Moment Method Analysis of Linearly Tapered Slot Antennas

by

Adnan Köksal

Low Loss Components for Switched Beam Radiometers

Final Report

NASA Grant No: NAG-1-943
Covering the period
Jan. 15, 1989 to Sept. 30, 1991

Co-Principal Investigators: Robert J. Trew
J. Frank Kauffman

Department of Electrical and Computer Engineering

North Carolina State University

Raleigh, NC

November 1992

Abstract

A Method of Moments (MoM) model for the analysis of the Linearly Tapered Slot Antenna (LTSA) is developed and implemented. The model employs an unequal size rectangular sectioning for conducting parts of the antenna. Piecewise sinusoidal basis functions are used for the expansion of conductor current. The effect of the dielectric is incorporated in the model by using equivalent volume polarization current density and solving the equivalent problem in free-space. The feed section of the antenna including the microstripline is handled rigorously in the MoM model by including slotline short-circuit and microstripline currents among the unknowns. Comparison with measurements is made to demonstrate the validity of the model for both the air case and the dielectric case. Validity of the model is also verified by extending the model to handle the analysis of the skew-plate antenna, and comparing the results to those of a skew-segmentation modeling results of the same structure and to available data in the literature. Variation of the radiation pattern for the air LTSA with length, height and taper angle is investigated and the results are tabulated. Numerical results for the effect of the dielectric thickness and permittivity are presented.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Problem Statement and Objectives

The main objective of this work is to develop a Moment Method Model for the radiation pattern characterization of single Linearly Tapered Slot Antennas (LTSA) in air or on a dielectric substrate. The geometry of the LTSA is shown in Figure 1.1. This characterization consists of:

- Finding the radiated far-fields of the antenna,
- Determining the E-Plane and H-Plane beamwidths and sidelobe levels,
- Determining the D-Plane beamwidth and cross polarization levels,

as antenna parameters length (L), height (H), taper angle (α), substrate thickness (d) and the relative substrate permittivity (ϵ_r) vary. The ranges of these parameters are:

$$0.25\lambda_0 \leq L \leq 5\lambda_0$$

$$0.25\lambda_0 \leq H \leq 3\lambda_0$$

$$2.5\,\text{deg} \leq \alpha \leq 9\,\text{deg}$$

$$0.01\lambda_0 \leq d \leq 0.1\lambda_0$$

$$1 \leq \epsilon_r \leq 10.5$$

where λ_0 is the free-space wavelength at the operating frequency.

The reason for these choices of parameter ranges will be explained in later sections.

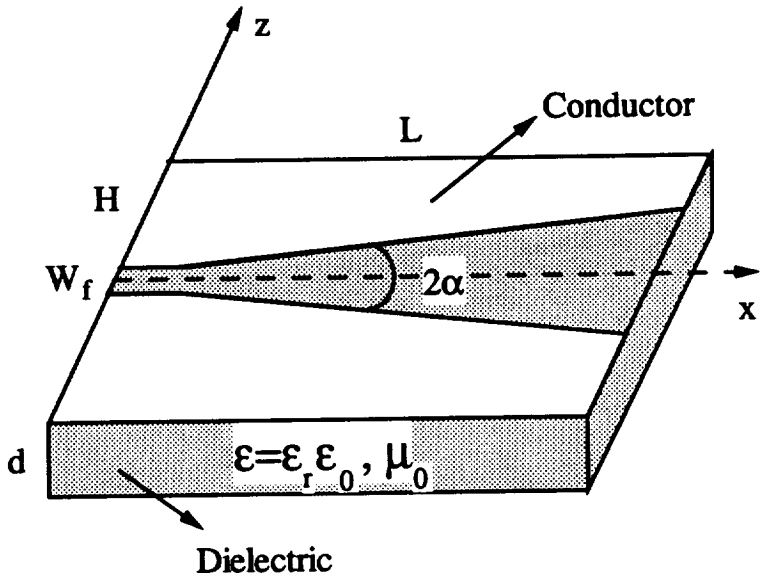


Figure 1.1: LTSA Geometry

The LTSA geometry which is shown in Figure 1.1, does not lend itself to analytical solution with the given parameter ranges. Therefore, a computer modeling scheme and a code are necessary to analyze the problem. This necessity imposes some further objectives or requirements on the solution method (modeling) and tool (computer code). These may be listed as follows:

- A good approximation to the real antenna geometry.
- Feasible computer storage and time requirements.

According to these requirements, the work is concentrated on the development of efficient modeling schemes for these type of problems and on reducing the central processing unit (CPU) time required for the computer code. A Method of Moments (MoM) code is developed for the analysis of LTSA's within the parameter ranges given.

1.2 Significance

An antenna is one of the most important components in all communication systems. With the development of radar during the 1940's, many new antennas have been introduced. The introduction of MoM to electromagnetics made the numerical analysis of many antennas possible. During the 1970's the developments in microstrip devices and circuits vastly increased the possibility of new antenna structures. Planar antennas enjoy the possibility of integration with the other parts of the system. In recent years, the new developments in the millimeter wave frequencies has increased the importance of planar antennas suitable for this frequency range. The tapered slot antenna is one likely candidate for both microwave and millimeter wave systems. It can be easily integrated to microstrip circuits with a microstrip-to-slotline transition. Its radiation characteristics are also promising. Initial studies on LTSA's were mostly

experimental. In recent years, some approximate analytical and numerical solutions for LTSA's have been developed, however the validity of the results are restricted by the choices of the antenna parameters or by the approximation of the real antenna geometry. Therefore, there is a need for better modeling and characterization of these antennas.

This work concentrates on rigorous computer modeling of single LTSA's with the use of MoM. The real antenna geometry is modeled closely for the first time, both the conducting parts of the antenna and the finite size dielectric region. Earlier modelings lacked accuracy in either modeling the conductor parts (by assuming a different shape of the conductors to ease the analysis), or in modeling the dielectric (by assuming the dielectric support infinite or assuming that it is very thin). The characterization of the LTSA will provide the researchers and designers in the field with better design guidelines in the range of the parameters given before. Also, the solution method is not unique to the problem, the analysis of similar structures may be carried out with a modification of the computer code for the particular problem.

1.3 Background

The Tapered Slot Antenna was introduced by Gibson [1]. He called it the Vivaldi Antenna. Since its introduction, its properties have been studied by many researchers in the field [2]-[12]. The first studies were mostly experimental [2, 3], dealing mostly

with the characterization of the antenna and derivation of some empirical design formulas. The usage of the antenna at millimeter wave frequencies as a feed array element [8], and the integration with the other elements of the system have also been studied [11, 12].

The first theoretical formulation for the radiation pattern of the antenna has been provided by R. Janaswamy [4, 6, 7]. In his work, the height of the LTSA (H in Figure 1.1) was assumed infinite. Assuming also infinitely long antennas, ($L \geq 3\lambda_0$), enabled him to approximate the fields by those of two infinite cones which can be analyzed analytically [13]. However, assuming infinite conducting parts for the antenna leads to incorrect predicted fields if one uses the free-space Green's function in the application of the Schelkunoff's equivalence principle [14]. Due to this reason, he approximated the effect of the finite length of the antenna by assuming a conducting half-space at the end of the antenna, and using the related Green's function.

In the analysis of LTSA's on a dielectric substrate the preceeding assumptions do not lead to an analytical solution due to the presence of the dielectric. Therefore, for this part of the analysis, he approximated the antenna taper step-wise, and solved the eigenvalue problem [15], to determine the aperture field distribution up to a multiplicative constant in each constant slot-line section [7]. Enforcing the power conservation principle for each junction yielded the field distribution which in turn was used to find the fields of the antenna with the same half-space Green's function. This analysis also assumes an infinitely long structure and the effect of the dielectric is

not taken into account when finding the radiated fields using the aperture distribution.

In this early work, it has been found experimentally that the radiation properties of the antenna improve as the antenna height gets smaller [4, 7]. However, the summarized analysis is not suitable for the solution of this problem. It also suffers from the treatment of the dielectric presence. In many applications, [2, 11, 12], the length of the antenna is less than $3\lambda_0$, further restricting the applicability of these results. Therefore, there has been a need to analyze the problem within parameter ranges given in Section 1.

Contemporary to our work, attempts have been made to model the antenna by the MoM [9, 10]. In this work, the antenna has been modeled by the skew-plate geometry shown in Figure 1.2. The shortcoming of this approximation is the assumption of differently shaped conductor edges in the distances comparable to the wavelength. Since the approximated geometry of the antenna has also been utilized in experimental models, good agreement with measurements has been obtained. Our results [16] predict different radiation patterns for the real geometry of the LTSA. Also the effect of the dielectric still needed better consideration, since only low-permittivity (ϵ_r), and very thin dielectric support has been analyzed [10].

In electromagnetic radiation and scattering problems there are two main approaches: Differential equation (DE) modeling and integral equation modeling [17]. Traditionally, DE models are used in bounded problems and IE models are employed

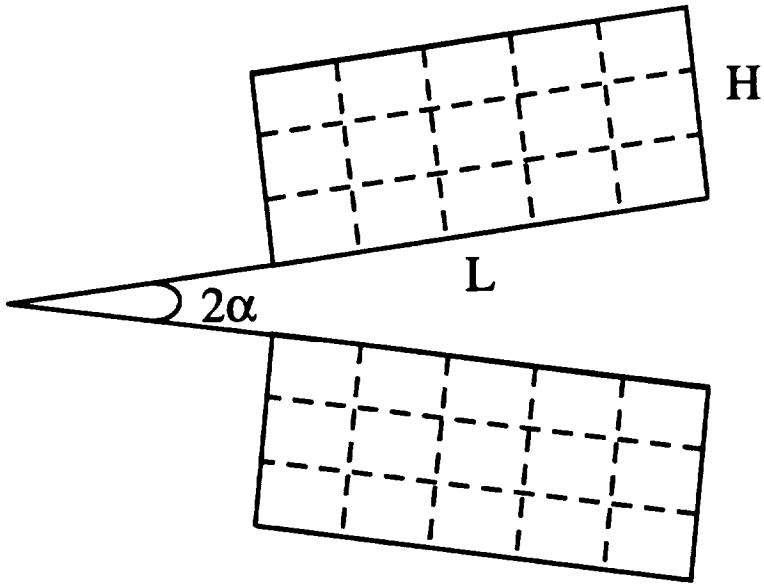


Figure 1.2: Skew-Plate antenna

in exterior radiation and scattering problems. In DE models, the problem region is divided into meshes and the unknown is approximated by a function in each cell. The satisfaction of the boundary conditions at each mesh boundary yields the unknowns related to the fields. In IE models, the integral equation describing the problem is obtained first, then, the unknown in the equation is expanded using some basis functions. Weighting of the IE with some weight functions converts the integral equations to a linear system of equations. Solution for the currents (or unknown in the problem) is obtained by standard matrix inversion or iterative techniques. By their nature, DE models are local and IE models are global, and as a result, the matrices obtained from DE models are large but sparse as opposed to the relatively small and dense matrices obtained from IE models. DE models are extended to radiation and scatter-

ing problems in unbounded regions by utilizing the "absorbtive boundary conditions" [18]. However, since large matrices are obtained using DE models and since the solution times are on the same order for both of the methods, we have preferred the IE modeling scheme and MoM formulation [19, 20].

In IE methods, another recent approach is the Conjugate Gradient Method [21, 22], which has been applied to dielectric scattering, scattering from conducting plates and wire antenna problems. However, it is not applicable to mixed bodies such as the dielectric supported LTSA. It has been applied to LTSA's in air and infinite arrays of LTSA's by Catedra et al [23].

1.4 Overview of Report

The report is organized as follows. Chapter 2 presents the formulation of generalized scattering or radiation from a coated dielectric body problem. In Chapter 3, the implementation of the method for the LTSA is explained. The modeling approach for the conducting and the dielectric parts of the antenna with the basis and test function choices for MoM formulation is given in this chapter. Possible excitation types for the antenna and the modeling of the source are also discussed in Chapter 3. In Chapter 4, the results and discussion are presented. In section 4.2, favorable comparison to available data in the literature and to experimental measurements is made to verify the computational results. The accuracy of the results is checked and

discussed. A parametric study of air LTSA's with changing L, H and α is given in section 4.3. Conclusions on the behavior of the radiation characteristics of the antenna with respect to these parameters are drawn. In section 4.4, the effect of the dielectric thickness and the permittivity on the radiation characteristics of the antenna are presented. The developed MoM code is explained briefly in Chapter 5. The performance study of the code is also given in this chapter. Finally, the conclusions and the suggestions for future research are presented in Chapter 6.

CHAPTER 2

FORMULATION OF THE METHOD

2.1 Introduction

Many important electromagnetic problems involve radiation and scattering from a dielectric body partially covered with a conductor. This general problem geometry is shown in Figure 2.1. In this figure, the problem is shown as a scattering problem where

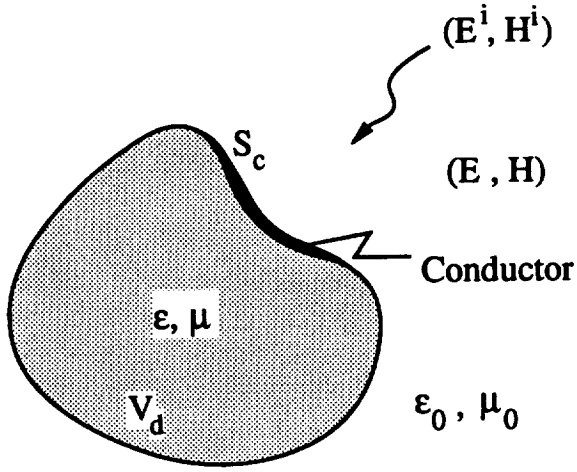


Figure 2.1: Scattering Problem Geometry

(E^i,H^i) is the incident field and (E,H), the total field in the presence of the scatterer, is the unknown. The location of the source for (E^i,H^i) is assumed to be at infinity so

that ($\mathbf{E}^i$,$\mathbf{H}^i$) is not effected by the presence of the scatterer. In radiation problems the same formulation as for the scattering case can be used with only a proper change of the interpretation of the incident field. In this type of problem, ($\mathbf{E}^i$,$\mathbf{H}^i$) is the field of the source ($\mathbf{J}_i$,$\mathbf{M}_i$) which is usually on or in the structure and assumed to be known or approximated. When attempting to solve the problem numerically, a suitable source model is chosen. As a result of this modeling, some parts of the source model should also be included as having unknown current distribution. This is the major difference of the radiation and scattering cases.

Whether it is a scattering or a radiation problem, the analytical solution of the total fields for Figure 2.1 is very difficult in most cases. When the geometry of the dielectric body is a canonical one such as a sphere or a slab extending to infinity, the specific Green's function can be derived in the frequency domain or a series representation of it may be obtained. However, the solution for the conductor parts of the structure is still very difficult and is usually carried out by a numerical approach [24] such as MoM. For example, when the dielectric body is a slab extending to infinity, the Green's function is easy to derive in the frequency domain [24]. However, in the solution of the unknown conductor currents by moment method, one encounters Sommerfeld integrals which are difficult to integrate numerically. When using series form for the Green's function, the slow convergence is a typical problem.

When the difficulties regarding the Green's function are considered and when the geometry of the particular problem does not allow these approaches, the only possible

way is to use numerical methods [17].

In order to solve the general problem of Figure 2.1 numerically, the governing integral equations for the conductor and dielectric regions are obtained. This is explained in section 2.2 These equations are then solved numerically using MoM. This procedure is detailed in section 2.3. In the remaining sections of this chapter, the dielectric body will be assumed to have the permeability, μ_0, of free-space, which is the case for most antenna problems.

2.2 Derivation of the Integral Equations

Referring to Figure 2.1, the conducting parts of the structure ,S_c, can be modeled by applying Schelkunoff's equivalence [14] principle. According to this principle, the total tangential fields determine the equivalent electric and magnetic surface current densities,

$$\mathbf{J}_s = \mathbf{n} \times \mathbf{H} \tag{2.1}$$

$$\begin{aligned} \mathbf{M}_s &= - \mathbf{n} \times \mathbf{E} \\ &= 0 \end{aligned} \tag{2.2}$$

which are introduced on the surfaces of the conductor; both bottom and top. Here, $\mathbf{n}$ is the unit outward normal to the conducting body. $\mathbf{M}_s$ is equated to zero in (2.2) since a perfect conductor assumption is made and on a perfect conductor the

tangential electric field is zero. When $\mathbf{J}_s$ and $\mathbf{M}_s$ are introduced on the conductor, the conductor can be removed and the currents $\mathbf{J}_s$, $\mathbf{M}_s$ can be considered to radiate in free-space [25]. If the conductor is very thin, equivalent currents $\mathbf{J}_s$ and $\mathbf{M}_s$ might be considered as the vector sum of the currents on the top and bottom surfaces. Throughout the analysis this assumption will be made for the conductor regions.

In the dielectric region, V_d, Maxwell's equations may be written as:

$$\nabla \times \mathbf{E} = -j\omega\mu_0\mathbf{H} \tag{2.3}$$

$$\begin{aligned} \nabla \times \mathbf{H} &= j\omega\epsilon\mathbf{E} \\ &= j\omega\epsilon_0\mathbf{E} + j\omega(\epsilon - \epsilon_0)\mathbf{E} \end{aligned} \tag{2.4}$$

Subsequently, (2.4) can be rewritten as,

$$\nabla \times \mathbf{H} = j\omega\epsilon_0\mathbf{E} + \mathbf{J}_e \tag{2.5}$$

where

$$\mathbf{J}_e = j\omega(\epsilon - \epsilon_0)\mathbf{E} \tag{2.6}$$

Equation (2.5) can be interpreted as a Maxwell's equation in free-space with a current source $\mathbf{J}_e$ located at the position of the dielectric part of the structure. Therefore, one can replace the dielectric region with the equivalent volume electric polarization current density $\mathbf{J}_e$ and consider the whole problem in free-space [20, 26, 27, 28, 29]. The equivalent problem is shown in Figure 2.2.

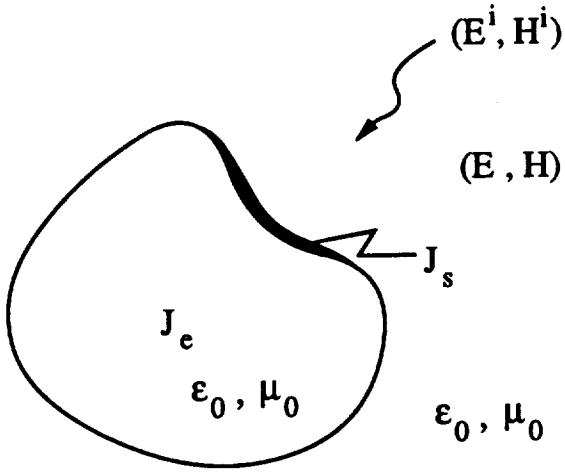


Figure 2.2: Equivalent Problem

On the conductor regions, the total tangential electric field intensity is zero. Therefore, the following equation must be satisfied on the conductor surfaces.

$$(\mathbf{E}^i + \mathbf{E}^s + \mathbf{E}^e)_{tan} = 0 \tag{2.7}$$

In equation (2.7), $\mathbf{E}^i$, $\mathbf{E}^s$, and $\mathbf{E}^e$ are the fields radiated by the sources $\mathbf{J}_i$, $\mathbf{J}_s$ and $\mathbf{J}_e$, respectively.

In the dielectric region, V_d, the condition

$$\begin{aligned} \mathbf{E}^s + \mathbf{E}^i + \mathbf{E}^e &= \mathbf{E} \\ &= \frac{\mathbf{J}_e}{j\omega(\epsilon - \epsilon_0)} \end{aligned} \tag{2.8}$$

should be satisfied.

Equation (2.8) is merely the statement of the equality of the total fields in which (2.6) has been used to obtain the relation with the equivalent polarization current density, $\mathbf{J}_e$, in the dielectric region.

All the fields produced by the equivalent sources and the source can be expressed in terms of the free-space dyadic Green's function as

$$\mathbf{E} = \int_V \mathbf{J} \cdot \overline{\overline{\mathbf{G}}} dV \tag{2.9}$$

where,

$$\overline{\overline{\mathbf{G}}} = (\bar{\bar{\mathbf{I}}} + \frac{1}{k_0^2}\nabla\nabla)g_0 \tag{2.10}$$

and,

$$g_0 = \frac{exp(-\jmath k_0 R)}{R}. \tag{2.11}$$

$\bar{\bar{\mathbf{I}}}$ is the unit dyad, R is the distance between the source and the field points, $R = |\mathbf{r} - \mathbf{r}'|$, where r and r' are position vectors to the field and source points, respectively, and k_0 is the free-space wavenumber. The ∇ operator operates on unprimed coordinates which are the field coordinates. For surface current densities, equations (2.9) through (2.11) still can be used with surface integrals over source current densities replacing the volume integrals.

As a result, equations (2.7) and (2.8) are the two integral equations that must be satisfied by the unknown conductor current density $\mathbf{J}_s$, and the equivalent volume polarization current density $\mathbf{J}_e$.

Equation (2.7) states that the total tangential electric field intensity on a conductor surface is zero. Therefore, if a test source $\mathbf{J}_m$ is placed in the conductor, its reaction [25] with all other sources, ($\mathbf{J}_i$, $\mathbf{J}_s$, $\mathbf{J}_e$), will be zero. In equation form this

can be written as:

$$\int_{S_c} \mathbf{J}_s \cdot \mathbf{E}^m dS + \int_{S_i} \mathbf{J}_i \cdot \mathbf{E}^m dS + \int_{V_d} \mathbf{J}_e \cdot \mathbf{E}^m dV =$$

$$\int_{S_m} \mathbf{J}_m \cdot \mathbf{E}^s dS + \int_{S_m} \mathbf{J}_m \cdot \mathbf{E}^i dS + \int_{S_m} \mathbf{J}_m \cdot \mathbf{E}^e dS = 0 \qquad (2.12)$$

where S_i and S_m are the regions in which $\mathbf{J}_i$ and $\mathbf{J}_m$ are nonzero. The field $\mathbf{E}^m$ is the field radiated by $\mathbf{J}_m$ in free-space.

Equation (2.12) is a reaction integral equation for the two unknown current densities $\mathbf{J}_s$ and $\mathbf{J}_e$. Satisfaction of this equation ensures that these currents have the proper reaction with a test source on the conductor surface. However, this does not insure the satisfaction of the field equality equation, (2.8), in the dielectric region. In order to incorporate the effect of the dielectric, we will multiply (2.8) by a vector weighting function $\mathbf{W}_m$, and integrate over V_d, to obtain,

$$\int_{V_d} (\mathbf{E}^s + \mathbf{E}^e - \frac{\mathbf{J}_e}{j\omega(\epsilon - \epsilon_0)}) \cdot \mathbf{W}_m dV = -\int_{V_d} \mathbf{E}^i \cdot \mathbf{W}_m dV \qquad (2.13)$$

Let us rewrite equations (2.12) and (2.13) as:

$$\int_{S_c} \mathbf{J}_s \cdot \mathbf{E}^m + \int_{V_d} \mathbf{J}_e \cdot \mathbf{E}^m = -\int_{S_i} \mathbf{J}_i \cdot \mathbf{E}^m$$

$$= -V_m \qquad (2.14)$$

$$\int_{V_d} (\mathbf{E}^s + \mathbf{E}^e - \frac{\mathbf{J}_e}{j\omega(\epsilon - \epsilon_0)}) \cdot \mathbf{W}_m dV = -\int_{V_d} \mathbf{E}^i \cdot \mathbf{W}_m dV$$

$$= -V_{m'} \qquad (2.15)$$

Equations (2.14) and (2.15) contain the unknown current densities in their kernels and are the governing reaction integral equations for the problem of Figure 2.1. These two coupled integral equations must be solved to find the unknown conductor and dielectric polarization current densities $\mathbf{J}_s$ and $\mathbf{J}_e$, respectively. The equations (2.14) and (2.15) must hold for any arbitrary test function $\mathbf{J}_m$ and $\mathbf{W}_m$. However, in order to solve these equations numerically with MoM, N distinct $\mathbf{J}_m$ and M distinct $\mathbf{W}_m$ will be used to reduce the equations to a square matrix equation, where N and M are the number of expansion functions for the conductor and dielectric regions, respectively.

It is worthwhile to mention here that, although the reaction concept is utilized to obtain (2.14), it is essentially an inner product of the integral equation (2.7) by the test functions $\mathbf{J}_m$ similar to the dielectric equation (2.8) and its inner product (2.15).

2.3 Solution of the Integral Equations by the Method of Moments

In order to solve equation (2.14) and (2.15) by MoM, the unknown currents are expanded as follows:

$$\mathbf{J}_s = \sum_{n=1}^{N} I_{sn}\mathbf{J}_{sn} \tag{2.16}$$

$$\mathbf{J}_e = \sum_{n=N+1}^{N+M} I_{en}\mathbf{J}_{en} \tag{2.17}$$

The total conductor surface current $\mathbf{J}_s$ is expanded by using the basis functions

$\mathbf{J}_{sn}$. In the dielectric, the volume polarization current density is similarly expanded using the basis functions $\mathbf{J}_{en}$. In (2.16) and (2.17) $\mathbf{J}_{sn}$ and $\mathbf{J}_{en}$ have known forms and I_{sn} and I_{en} are the unknown multiplicative constants to be determined. In general, since the conductor current is a surface current density, $\mathbf{J}_{sn}$ should contain two orthogonal components, whereas $\mathbf{J}_{en}$ is a volume current density and should contain three.

Substituting (2.16) and (2.17) into (2.14) and changing the order of integration and summation gives,

$$\sum_{n=1}^{N} I_{sn}\left(\int_{S_c} \mathbf{J}_{sn} \cdot \mathbf{E}^m dS\right) + \sum_{n=N+1}^{N+M} I_{en}\left(\int_{V_d} \mathbf{J}_{en} \cdot \mathbf{E}^m dV\right) = -V_m$$

$$= -\int_{S_i} \mathbf{J}_i \cdot \mathbf{E}^m dS \qquad (2.18)$$

In (2.18), since $\mathbf{J}_{sn}$,$\mathbf{J}_{en}$ and $\mathbf{E}^m$ are known, the integrals may be evaluated leaving a linear equation in $N+M$ unknowns. Since $\mathbf{E}^m$ is the field of the test sources which are placed on the conductor, using N test sources in (2.18) gives N equations in $N+M$ unknowns.

Similarly, when (2.16) and (2.17) are substituted into (2.15) we obtain,

$$\sum_{n=1}^{N} I_{sn}\left(\int_{V_d} \mathbf{E}^{sn} \cdot \mathbf{W}_m dV\right) +$$

$$\sum_{n=N+1}^{N+M} I_{en}\left(\int_{V_d} \left(\mathbf{E}^{en} - \frac{\mathbf{J}_{en}}{j\omega(\epsilon - \epsilon_0)}\right) \cdot \mathbf{W}_m dV\right) = -V_{m'}$$

$$= -\int_{V_d} \mathbf{E}^i \cdot \mathbf{W}_m dV \qquad (2.19)$$

where $\mathbf{E}^{sn}$ and $\mathbf{E}^{vn}$ are the fields produced by the surface and the volumetric basis functions (currents), $\mathbf{J}_{sn}$ and $\mathbf{J}_{vn}$, respectively. When M weighting functions are used in (2.19), M equations in $N + M$ unknowns result. Together with those obtained from (2.18) a square matrix of order $N + M$ is obtained.

Equations (2.18) and (2.19) represent a linear system of equations which can be written compactly,

$$\sum_{n=1}^{N+M} I_n Z_{mn} = V_m \quad \text{for } m = 1, \ldots, N + M \tag{2.20}$$

or in the matrix form as,

$$\mathbf{ZI} = \mathbf{V} \tag{2.21}$$

where, Z is the square impedance matrix, I is the current vector, and V is the excitation or voltage vector, and,

$$I_n = \begin{cases} I_{sn} & \text{if } n = 1, \cdots, N \\ I_{vn} & \text{if } n = N + 1, \cdots, N + M \end{cases} \tag{2.22}$$

$$V_m = \begin{cases} -\int_{S_i} \mathbf{J}_i \cdot \mathbf{E}^m dV & \text{if } m = 1, \cdots, N \\ -\int_{V_d} \mathbf{E}^i \cdot \mathbf{W}_m dV & \text{if } m = N + 1, \cdots, N + M \end{cases} \tag{2.23}$$

After the matrix elements, z_{ij}, are calculated using numerical integration, the unknown current coefficients are solved by standard inversion or iteration procedures.

Until this point the method is general in the sense that, neither the basis functions for the conductor and the dielectric, nor the testing (weighting) functions for them

are specified. The difficulty or the complexity of the matrix element evaluation and the computation time for them are heavily influenced by these choices. The chosen basis and testing functions and their impact on the implementation of the method for LTSA's will be explained in the next chapter.

CHAPTER 3

IMPLEMENTATION OF THE METHOD FOR LINEARLY TAPERED SLOT ANTENNA

3.1 Introduction

In this chapter, the implementation of the method explained in chapter 2 will be given. In order to apply the formulation of the previous chapter to the antenna geometry of Figure 3.1, the conducting parts of the antenna should be approximated by a surface modeling scheme. The definitions of the unknown currents on the conducting surfaces

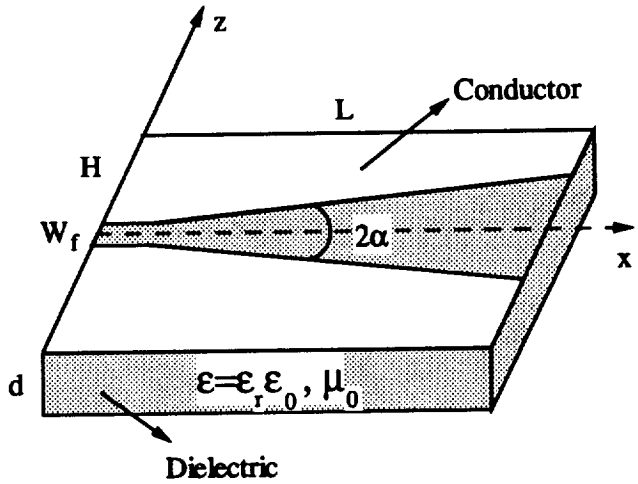


Figure 3.1: LTSA Geometry

will complete this part of the analysis. The next step is to approximate the dielectric geometry and define the unknown currents for the dielectric region. When the test functions for both the conducting and the dielectric parts are determined, the matrix elements can be calculated.

The approximation of the conducting and dielectric parts also involves the modeling of the source. Depending on the source modeling, unknowns related to the source may be included in the matrix equation.

After the matrix equation is obtained by calculating the matrix elements and the right-hand side vector, the solution of this equation gives the unknowns. Once the unknown currents are calculated, any necessary information of the antenna such as the far-field radiation pattern, field distribution in the dielectric, input impedance, can be calculated easily.

3.2 Conductor Modeling

Possible choices for modeling the conductor surfaces are triangular sectioning [30, 31], polygonal plate modeling [32, 33, 34], or a combination of these with rectangular sectioning [35, 36]. As mentioned in chapter 2, the complexity of the matrix element calculation depends on this choice. Although triangular and polygonal plate modeling schemes are better in conformity to the surface than rectangular sectioning,the number of integrations involved in calculating the matrix elements is larger. Triangular

sectioning is best suited for modeling the LTSA geometry, however the integrations in the matrix element calculations have to be carried out on a triangular domain which is costly and difficult to do numerically. Polygonal plate sectioning is also suited to modeling the LTSA geometry. The difficulty in matrix element calculation however is worse than both the triangular and rectangular sections. The matrix element calculation integrations are four-fold in this case. Rectangular sectioning gives the simplest expressions for the matrix elements and is the least costly in terms of the computation time. Hence, whenever applicable, rectangular sectioning offers simplicity and computational savings. Referring to Figure 3.1, the range of the taper angle for useful antennas was determined earlier [4] to be less than 9 degrees. This small taper angle allows a good approximation with the unequally sized rectangular segmentation of Figure 3.2.

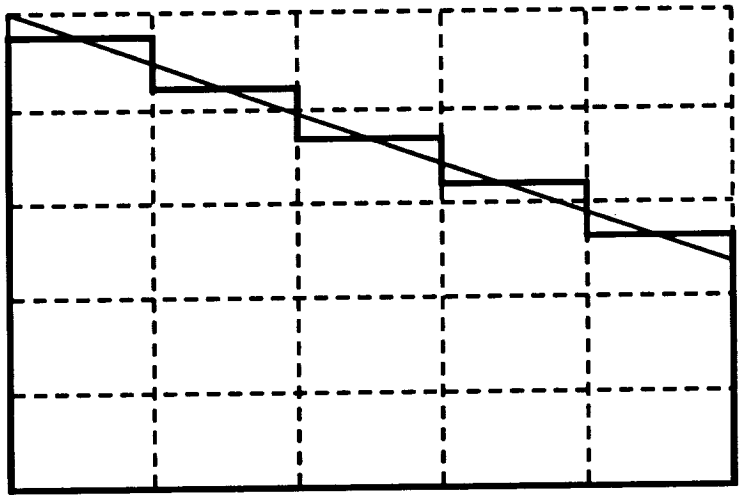


Figure 3.2: Unequal Size Rectangular Sectioning

3.3 Conductor Basis and Test Functions

The conductor current basis fuctions are chosen as overlapping piecewise sinusoidal functions. The current on the ith segment of the conductor consists of two monopoles. The current starts on the segment i and extends to the next segment. Each segment carries a monopole current which has components J_{z_i} and J_{x_i} defined by,

$$J_{z_i} = \mathbf{a}_z \frac{1}{2w_i} \left\{ \frac{I_{1i} \sin[k_0(h_i - z_i)] + I_{2i} \sin(k_0 z_i)}{\sin(k_0 h_i)} \right\} \tag{3.1}$$

$$J_{x_i} = \mathbf{a}_x \frac{1}{h_{i+1}} \left\{ \frac{I_{1i} \sin[k_0(w_i - x_i)] + I_{2i} \sin[k_0(w_i + x_i)]}{\sin(2k_0 w_2)} \right\} \tag{3.2}$$

where $\mathbf{a}_z$ and $\mathbf{a}_x$ are the unit vectors in the directions of z and x, respectively, $2\omega_i$ is the segment width, and h_i is the segment height. I_{1i} and I_{2i} are the terminal currents, and take the values either 0 or 1. z_i and x_i are the local coordinate variables of the monopole measured from the bottom and the center of the monopole, respectively (See Figures 3.4 and 3.5). The x component of the monopole current extends h_{i+1} along z, to provide current overlap for the successive unequal-size rectangular sections. These monopole currents are piecewise sinusoidal in the current direction and constant transverse to it, and are the same as those in [25].

The combination of the two neighboring segment currents creates one unknown for the conductor. This combination and the resulting current distribution is shown in Figure 3.3. As seen from Figure 3.3, the conductor currents are continuous in the current direction since every surface dipole current overlaps a neighboring one.

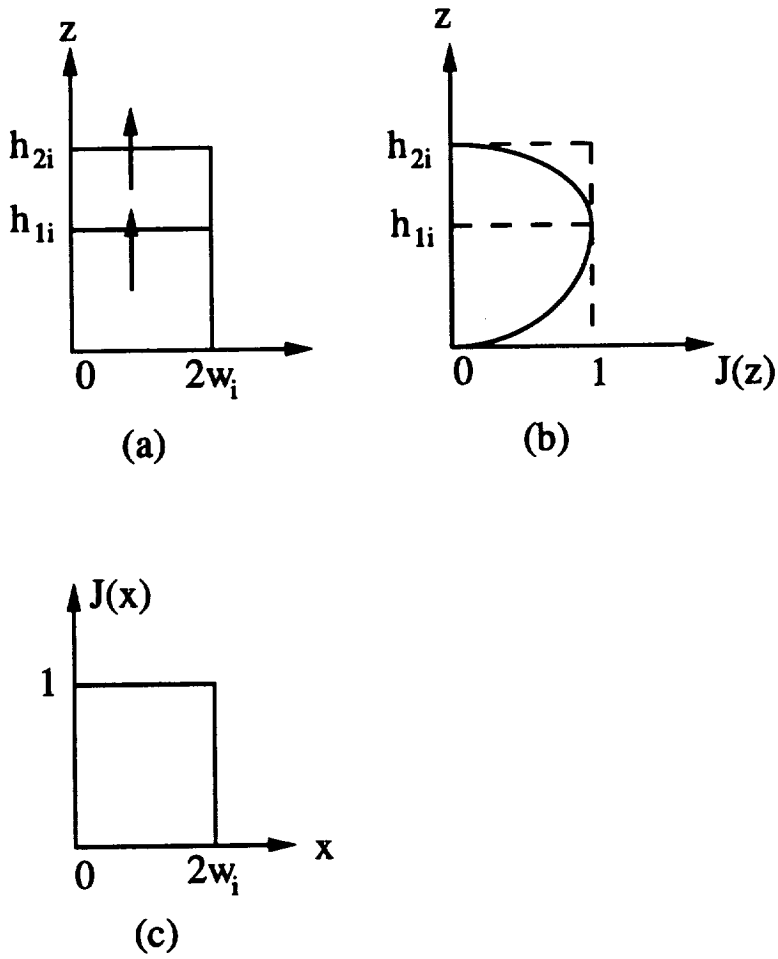


Figure 3.3: Piecewise Sinusoidal Conductor Currents a) Segment geometry b) Distribution in Current Direction c) Distribution in Perpendicular Direction

The current is a constant with respect to the coordinate perpendicular to the current direction. The continuity of the current is a desired property for two reasons. Firstly, the current on the conductor is continuous, therefore, using continuous basis functions allows a good approximation, especially where the current is rapidly varying. Secondly, a discontinuous current approximation creates line charges where the current is discontinuous. Since this would be a fictitious line charge that shouldn't

be present it might lead to erroneous results for antenna currents and therefore other calculated results.

The test functions to complete the application of (2.18) are chosen the same as the basis functions. This is called Galerkin formulation. For this specific choice of basis and test functions, the matrix elements for the conductor-conductor interactions become mutual impedances between the respective currents. These mutual impedances are obtained by summing up the four monopole-to-monopole mutual impedances [25]. With the rectangular sectioning and the defined current distributions, there are only two types of monopole-to-monopole mutual impedance calculations. These are parallel and perpendicular cases which are shown in Figure 3.4 and Figure 3.5. With this choice of rectangular sectioning and current definition the monopole-to-monopole mutual impedances of Figure 3.4 and Figure 3.5 were earlier calculated by integrating the mutual impedance of line currents [36, 37, 38, 39] over the monopole surfaces. The simplest formula for the parallel case is reported in [40]. For the perpendicular case, a one dimensional integral formula for the mutual impedance between a dipole and a monopole is reported in [41]. None of these earlier formulations is valid for unequally sized parallel or perpendicular monopole-to-monopole interactions. The direct integration for the surface currents leads to faster and easier evaluation of matrix elements even in the general case of unequal size segments. Simple formulas for the mutual impedances of the two cases shown in Figure 3.4 and Figure 3.5 have been derived and reported in [42]. The derivations and the resulting expressions will

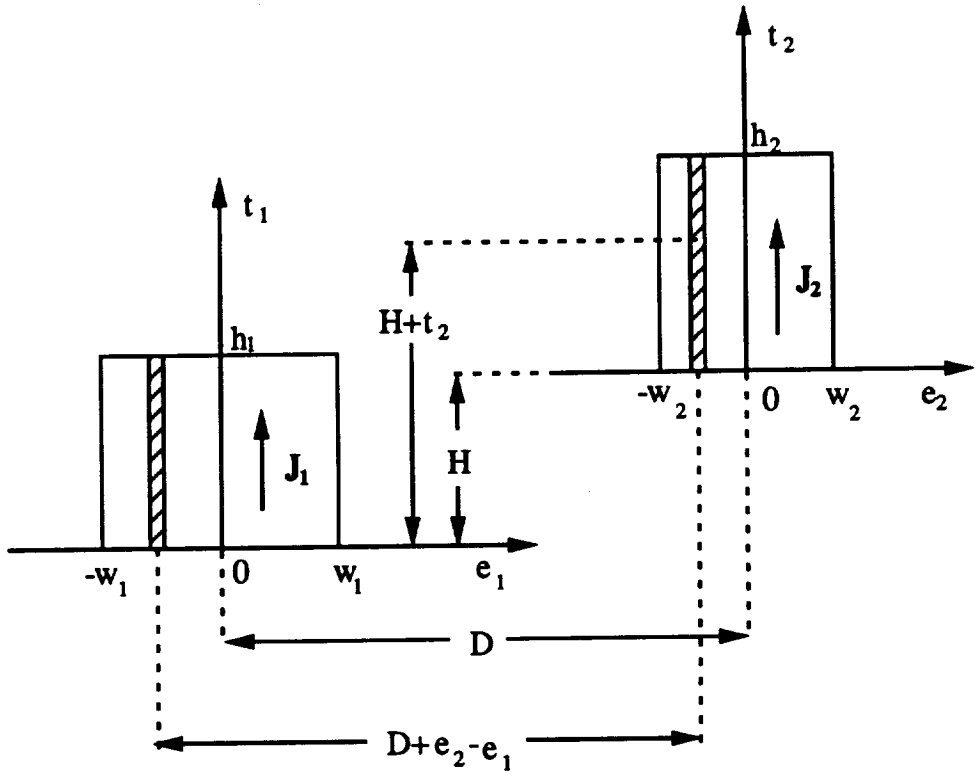


Figure 3.4: Parallel Monopoles

not be repeated here, however, it is worthwhile to mention that the parallel case contains only one-dimensional integrals and the perpendicular case is in closed form, containing exponential integrals only. It is these simple formulas that led to the choice of unequal size rectangular sectioning of the conductor. Tremendous savings of computation time makes possible the use of finer grid sizes and therefore better approximations of the antenna geometry. Another resulting benefit of the rectangular sectioning is the increased symmetry that further reduces the computation time. In large method of moment calculations it is important to consider the symmetries and eliminate the unnecessary computations. In summary, compared to triangular

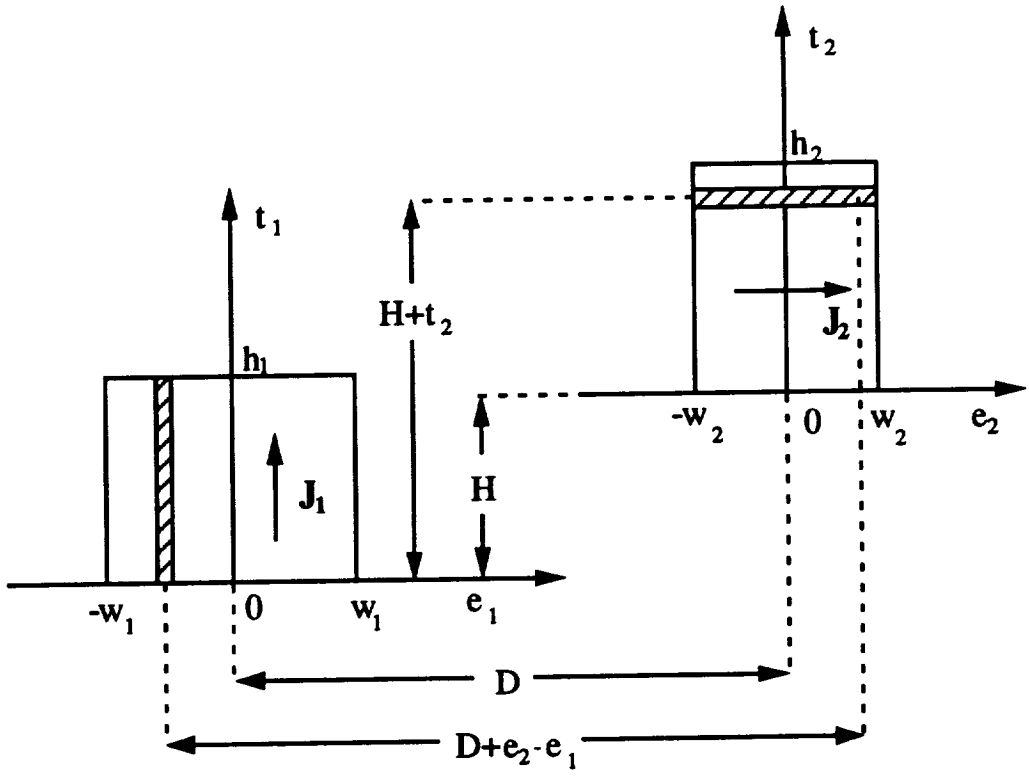


Figure 3.5: Perpendicular Monopoles

and polygonal plate modeling, rectangular sectioning is more efficient for the LTSA geometry.

3.4 Dielectric Modeling

The dielectric region of the LTSA of Figure 3.1 is a rectangular slab with a thickness *d*. Therefore, any sectioning scheme would easily conform to its geometry. The most common modeling technique for the slab geometry that has extensively been used in the earlier work is cubical sections [26, 27, 28, 29]. For arbitrary dielectric shapes,

tetrahedron modeling has been employed [43] . Since cubical sectioning has been successfully used in the modeling of similar antenna structures, it has been employed in this work (See Figure 3.6).

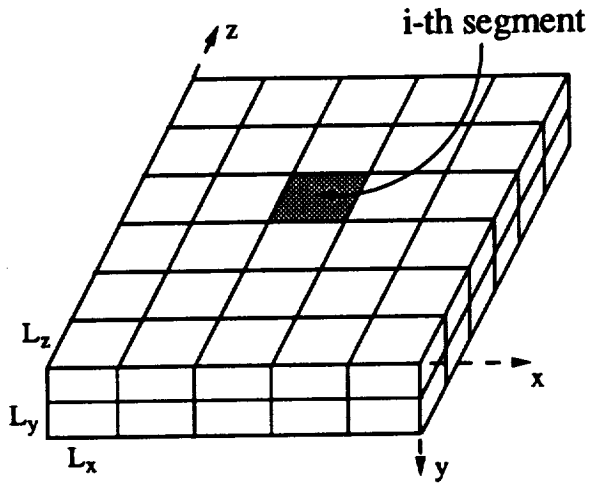


Figure 3.6: Dielectric Segmentation

3.5 Dielectric Basis and Test Functions

In the dielectric region, pulse basis functions are employed which are suitable for the segmentation of Figure 3.6. Therefore, the polarization current density in the dielectric region is expanded as

$$\mathbf{J}_{ei}(x,y,z) = \frac{\alpha_i}{L_y L_z}\mathbf{a}_x + \frac{\beta_i}{L_x L_z}\mathbf{a}_y + \frac{\gamma_i}{L_x L_y}\mathbf{a}_z \tag{3.3}$$

if (x,y,z) is in the i-th cell , (Figure 3.6), and zero if outside.

In the analysis of microstrip antennas, the assumption of infinite dielectric is

usually employed [44, 45, 46]. Since the finite dimensions of the dielectric are ignored in this assumption and since Sommerfeld-type integral evaluations are necessary for the calculation of the matrix elements, this approach is unsuitable for the LTSA analysis. In the analysis of scattering by homogeneous dielectrics, surface current formulations have also been employed [47, 48], which is also applicable to the LTSA problem. However, the case study for a sample LTSA has revealed that the surface current formulation would lead to larger matrix sizes and more difficult numerical integrations for the matrix element computations compared to the volumetric pulse expansion given in equation (3.3). The simplicity of volumetric pulse functions in terms of integration complexity was also the reason why overlapping triangular or sinusoidal basis functions were not preferred.

The expansion of the dielectric volume polarization current density by (3.3) satisfies the criteria that the divergence of the electric field intensity inside a homogeneous dielectric region is zero [29]. However, it introduces surface polarization charges on the faces of the volumetric pulses. The effect of these surface charges has been found to be negligible for the LTSA modeling.

The same modeling scheme for the dielectric regions has been successfully applied to dielectric scattering problems and wire antenna problems [28]. Recently, it has also been applied to the analysis of microstrip antennas [29].

The test functions for the dielectric region are chosen as delta functions located

at the center of each cell and directed in $\mathbf{a}_x$, $\mathbf{a}_y$ and $\mathbf{a}_z$ directions. When employed in (2.19), this test function choice results in the field equation (2.8), which is merely a statement of the equality of the total fields at the center of each dielectric segment and for each component of the electric field intensity in $\mathbf{a}_x$, $\mathbf{a}_y$ and $\mathbf{a}_z$ directions.

The reason for the choice of this test function is basically its simplicity. Other test functions would introduce additional complexity in addition to already numerically difficult field calculation for three dimensional sources, especially in the source region itself [49].

3.6 Source Modeling

Good source modeling in the LTSA analysis is very important because of the fact that the source can contribute to the radiation pattern appreciably. The reason for the source contribution is the openness of the feeding structures. The source region of the LTSA may differ appreciably according to the mode in which the antenna is being used or according to the feeding structure that is employed. Two of the most commonly used source configurations are shown in Figure 3.7. When the antenna is used in the receiving mode, power can be picked up easily by a detector diode soldered across the apex of the antenna as shown in part a) of Figure 3.7

The second source configuration employs a microstripline-to-slotline transition [50] as shown in part b) of Figure 3.7. In this figure, L_i is the input transition

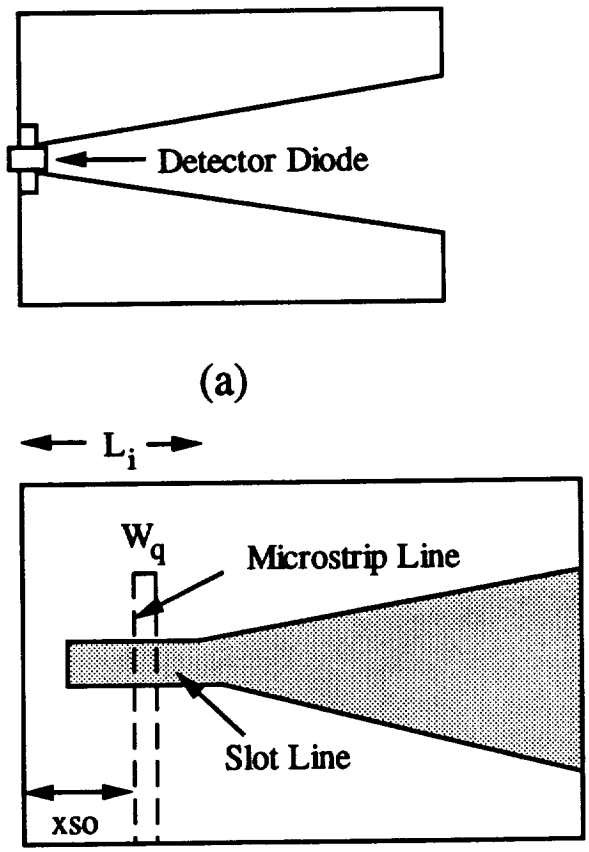


Figure 3.7: Possible Source Configurations of the LTSA a) Receiving mode with a detector diode b) LTSA with a microstripline to slotline transition

length, $xs0$ is the distance of the microstripline to the antenna edge and W_q is the width of the microstripline. In this configuration power is delivered to the slotline with a microstripline which extends $\lambda_{ms}/4$ past the slotline edge, where λ_{ms} is the microstrip wavelength. Slotline, on the other hand, is short-circuited $\lambda_{sl}/4$ away from the microstripline edge, λ_{sl} being the slotline wavelength. This configuration creates a resonant structure with a very good voltage standing wave ratio over narrow

bandwidths [50]. The bandwidth can be increased by using matching circuitry on the microstripline, while impedance matching the latter to the other parts of the circuit at the same time.

Different source configurations of the LTSA must be modeled differently. For the diode reception mode, the diode is modeled as a dipole with a delta gap generator at its center [20]. When the dipole thickness is made equal to the thickness of the detector diode, it has been shown earlier that this source modeling yields good results [10].

For the input configuration of Figure 3.7, the microstrip radiation is initially modeled by an infinitesimal current element at the center of the slotline and the other parts of the antenna are approximated by rectangular sectioning, including the short circuit for the slot line. This has been accomplished by introducing another current at the exact short circuit location. The length of the short circuit current is made greater than the slot line width so as to make the current flow between the lower and upper parts of the antenna. It is found that the current element modeling for the microstripline gives very satisfactory results for the co-polar radiation characteristics of the antenna. However, it cannot predict the correct level of cross polarized radiation in the boresight direction. Therefore, a third source model is developed which accounts for the microstripline rigorously, by defining currents on the microstrip as well. The feed point of the microstrip is approximated by a current element in y direction, which avoids the difficulty of introducing the connection mode

currents between the microstripline and the antenna plates. This third model is found satisfactory for determining the radiation characteristics of the antenna, as will be seen in Chapter 4.

3.7 Evaluation of the Matrix Equation

The choices of the conductor and dielectric basis and test functions of sections 3.3 and 3.5 determine how the matrix elements will be computed. Assuming N unknowns for the conducting parts and S segments for the dielectric region, there will be $3S$ unknowns in the dielectric. Therefore, $3S = M$ of (2.19). If we consider Z as,

$$\mathbf{Z} = \begin{bmatrix} A_{N\times N} & B_{N\times M} \\ C_{M\times N} & D_{M\times M} \end{bmatrix} \tag{3.4}$$

A and **B** submatrices will have elements which are calculated by reaction integrals. For currents $\mathbf{J}_i$ and $\mathbf{J}_j$ the reaction integral is given by

$$\begin{aligned} Z_{ij} &= \int_{S_j} \mathbf{E}^i \cdot \mathbf{J}_j \, dS \\ &= \int_{S_i} \mathbf{E}^j \cdot \mathbf{J}_i \, dS \end{aligned} \tag{3.5}$$

where $\mathbf{E}^i$ and $\mathbf{E}^j$ are the fields radiated by $\mathbf{J}_i$ and $\mathbf{J}_j$, respectively, and S_i and S_j are the regions where $\mathbf{J}_i$ and $\mathbf{J}_j$ are nonzero. Submatrix **A** consists of the conductor-conductor interactions. Its elements are calculated by the reaction integrals between the conductor currents only. The calculation of these elements is explained in Section

3.3 since it is related to the geometry approximation directly. These elements are calculated by using the expressions reported in [42].

Since delta functions are chosen as the testing functions in the dielectric region, C and D submatrices are comprised of the fields radiated by the conductor currents and the dielectric polarization currents respectively at the center of each segment. Evaluation of the C submatrix elements is carried out using field equations for piecewise sinusoidal line sources [14]. The field of any conductor current can be found by adding the fields of the corresponding monopoles that makes up the whole current. Referring to Figure 3.1, conductor currents can be in x or z directions only. For a current in z direction, the components of the electric field intensity, Z_{CD_s}, where s stands for x, y or z component, can be found as

$$Z_{CD_x} = \sum_{i=1}^{2} C_i \sum_{m=1}^{2} \int_{-k_0\omega_i}^{k_0\omega_i} \alpha_m \frac{(k_0D - x)(k_0L + \beta_m)}{[(k_0D - x)^2 + (k_0H)^2]} \frac{exp(-\jmath R_m)}{R_m} \tag{3.6}$$

$$Z_{CD_y} = \sum_{i=1}^{2} C_i \sum_{m=1}^{2} \int_{-k_0\omega_i}^{k_0\omega_i} \alpha_m \frac{k_0H(k_0L + \beta_m)}{[(k_0D - x)^2 + (k_0H)^2]} \frac{exp(-\jmath R_m)}{R_m} \tag{3.7}$$

$$Z_{CD_z} = \sum_{i=1}^{2} -C_i \sum_{m=1}^{2} \int_{-k_0\omega_i}^{k_0\omega_i} \alpha_m \frac{exp(-\jmath R_m)}{R_m} \tag{3.8}$$

where D, H and L are the distances between the center point of the monopole i and the center of the dielectric cell, in x, y and z directions, respectively, as shown in Figure 3.8. The various other parameters are

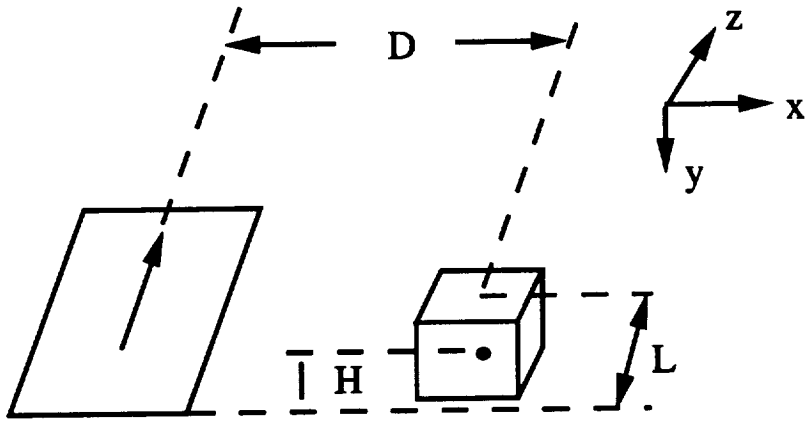


Figure 3.8: Geometry of conductor and dielectric currents

$$
\begin{aligned}
C_i &= \frac{-j30k_0}{2k_0\omega_i \sin(k_0 h_i)} \\
\alpha_1 &= I_{2i}\cos(k_0 h_i) - I_{1i} \\
\alpha_2 &= I_{2i} - I_{1i}\cos(k_0 h_i) \\
\beta_1 &= -k_0 h_i \\
\beta_2 &= 0 \\
R_m &= [(k_0 L + \beta_m)^2 + (k_0 D - x)^2 + (k_0 H)^2]^{1/2} \ .
\end{aligned}
$$

The fields of the x-directed currents can be found using equations (3.6-3.8) with a coordinate rotation.

Returning to the submatrix **B**, and considering that its elements are the reaction integrals between the conductor currents and the dielectric currents, the following approach is applied in their calculation. The dielectric cell is divided into smaller cubical sections and the fields of the conductor current is calculated at the center of

each small section. Multiplication of these fields with the volume of the subdivision and summation over the cell gives a good approximation to the elements of the B submatrix. Therefore, if the dielectric cell has dimensions L_x, L_y and L_z and if it is divided into n_x, n_y and n_z segments in x, y and z directions, respectively, the reaction, Z_{DC}, of a z directed conductor current on a dielectric current becomes

$$Z_{DC_s} = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \sum_{k=1}^{n_z} -Z_{CD_{s_{ijk}}} J_{c_s} \frac{L_x L_y L_z}{n_x n_y n_z} \tag{3.9}$$

where $Z_{CD_{s_{ijk}}}$ is the field of the conductor current at the center of the subdivision ijk, and s represents x, y or z directed dielectric current. The interactions between the x-directed conductor and dielectric currents can be found by using (3.9) with a coordinate rotation for the conductor current. This approximation to the dielectric-conductor interaction submatrix B is justified because of the fact that MoM usually yields diagonally dominant impedance matrices. Since the elements of B are off-diagonal, the results will not be as sensitive to errors in the calculation of these elements compared to the ones in diagonal elements of A and B.

The elements of the submatrix D are directly calculated using equations (2.9-2.11). When calculating the field of a cell current in itself, the singularity in the expression is extracted as suggested in [49]. For the off-diagonal elements of D, the same type of approximation as (3.9) is made to reduce the computation time.

The voltage vector calculation also follows the same approach. The first N elements are the reactions of the source and the conductor expansion currents, the

remaining $M = 3S$ are the fields radiated by the assumed source distribution at the center of the dielectric segments.

All of the numerical integrations for the evaluation of the matrix elements are carried out using Gauss-Chebychev quadratures which are suitable for oscillatory kernels.

3.8 Solution of the Matrix Equation

The matrix equation,

$$\mathbf{ZI} = \mathbf{V} \tag{3.10}$$

is solved by using the Conjugate Gradient (CG) method [51, 52, 53]. CG method is an iterative conjugate direction method. In Conjugate Direction method the error functional is minimized successively in the directions of a set of Z-orthogonal vectors. A set of vectors $\mathbf{P}_n$, $n = 1, 2, \ldots, (N+M)$ is said to be Z-orthogonal (or Z-conjugate if they satisfy

$$< \mathbf{ZP}_i \, , \mathbf{P}_j^* >= 0 \quad \text{for } i \neq j \tag{3.11}$$

where * denotes the conjugate.

In conjugate direction methods, at each iteration, $\mathbf{I}_n + \alpha_n \mathbf{P}_n$ is minimized where $\mathbf{P}_n$ lies on the $(N + M - 1)$ dimensional hyperplane

$$< \mathbf{P}_n^* \, , \mathbf{ZI} - \mathbf{V} >= 0 \tag{3.12}$$

whose normal is ZP. The conjugate direction methods differ in the way $\mathbf{P}_n$ are obtained. When the vectors $\mathbf{P}_n$ are obtained by Z-orthogonalization of the residual vectors, $\mathbf{R}_n$, which will be defined subsequently, a CG method results.

The CG method is applicable to Hermitian matrices (operators). In the dielectric supported LTSA case the matrix Z is not symmetric and hence, CG method cannot be applied to (3.10) directly [51]. In order to satisfy the symmetry and the positive definite requirements for Z, both sides of (3.10) can be multiplied by $\mathbf{Z}^T$, where ^{T} denotes transpose conjugate. The CG algorithm can then be applied to the transformed equation without actually forming the product $\mathbf{Z}^T\mathbf{Z}$ [54].

For an initial guess $\mathbf{I}_0$, the CG method starts by evaluating,

$$\begin{aligned} \mathbf{R}_0 &= \mathbf{V} - \mathbf{Z}\mathbf{I}_0 \\ \mathbf{P}_0 &= \mathbf{Z}^T\mathbf{R}_0 \end{aligned} \tag{3.13}$$

and then develops each successive approximation by,

$$\mathbf{I}_{n+1} = \mathbf{I}_n + \alpha_n \mathbf{P}_n \tag{3.14}$$

where

$$\alpha_n = \frac{||\mathbf{Z}^T\mathbf{R}_n||^2}{||\mathbf{Z}\mathbf{P}_n||} \,. \tag{3.15}$$

The residual vectors are generated as

$$\mathbf{R}_{n+1} = \mathbf{R}_n - \alpha_n \mathbf{Z}\mathbf{P}_n \,. \tag{3.16}$$

The direction vectors at each iteration are obtained as

$$\mathbf{P}_{n+1} = \mathbf{Z}^T\mathbf{R}_{n+1} + \beta_n\mathbf{P}_n \tag{3.17}$$

where

$$\beta_n = \frac{||\mathbf{Z}^T\mathbf{R}_{n+1}||^2}{||\mathbf{Z}^T\mathbf{R}_n||}. \tag{3.18}$$

This algorithm minimizes the norm of the residual, $||\mathbf{R}_n||$, at each iteration. The iterations are terminated when the error norm, $||\mathbf{R}_n||$ is less than some ratio of the initial error norm, $||\mathbf{R}_0||$. The initial guess in this work is taken as zero vector in all computed results. Therefore, $||\mathbf{R}_0|| = \mathbf{V}$. When $||\mathbf{R}_n|| \leq 10^{-4}||\mathbf{R}_0||$, the iterations are terminated. Different tolerance values (10^{-5}, 10^{-6}) have also been employed to check the sensitivity of the results around the solution with the same input data. The solutions obtained with smaller tolerances are nearly identical in terms of the radiation pattern and current distribution, and hence, 10^{-4} is used in further results.

CG method has many nice features that make it useful in the solution of large linear systems of equations such as the one obtained in the LTSA analysis. Some of these which might explain the preference of CG iteration over the direct methods can be outlined as [54, 55]:

- The method is highly insensitive to the initial guess $\mathbf{I}_0$. As mentioned earlier, $\mathbf{I}_0 = 0$ is chosen in all of the computations involved in this work, with no difficulty in obtaining the solution.

- The number of iterations required for CG method is equal to the number of distinct eigenvalues of the matrix **Z** [54]. This property makes CG method especially useful for large MoM applications, since in most of the cases, the eigenvalues of **Z** are closely spaced. Actually, it is this property that favors CG method over direct methods since CG method lowers its computational cost with closely spaced eigenvalues. In order for CG method to be less costly compared to Gaussian Elimination, the number of iterations should be less than $(N+M)/3$, where $(N+M)$ is the size of the matrix **Z**. Most of the results in this work are obtained with number of iterations less than this number.

- The convergence of CG method is $1/K$ quadratic [54], assuming that the solution is reached in K iterations. The reason for this rate definition of Sarkar [54] is that the algorithm requires K steps to achieve the effect of one step of a method with a true quadratic convergence rate.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

In this chapter, the results of the analysis of LTSA's in air or with a dielectric support will be given. Verification of the computed results and comparison to experimental measurements are given section 4.2. A parametric study of air LTSA's is explained in Section 4.3. Finally, the computed results for dielectric LTSA's are presented in Section 4.4, where the effect of the dielectric thickness and permittivity are investigated.

The radiation patterns of the antenna in the E-Plane , H-Plane and the D-Plane are used throughout this chapter. With the coordinate system of Figure 3.1 for the antenna configuration, E-Plane of the antenna coincides with the $x-z$ Plane, whereas H-Plane is the $x-y$ plane, as shown in Figure 4.1 and Figure 4.2, respectively. The D-Plane is a diagonal plane located at 45 degrees to the E and H-Planes. The radiated patterns are measured and computed for co-polar and cross-polar components. The co-polar component of the field of the antenna is the θ component for both principal

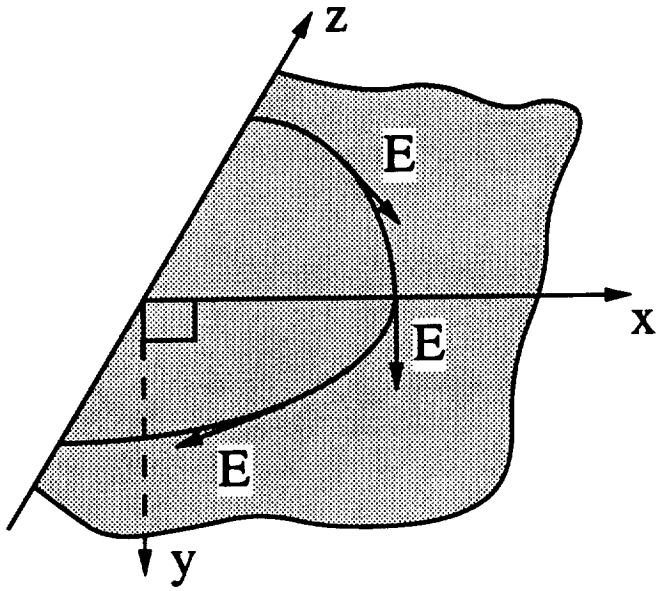


Figure 4.1: E-Plane of the LTSA

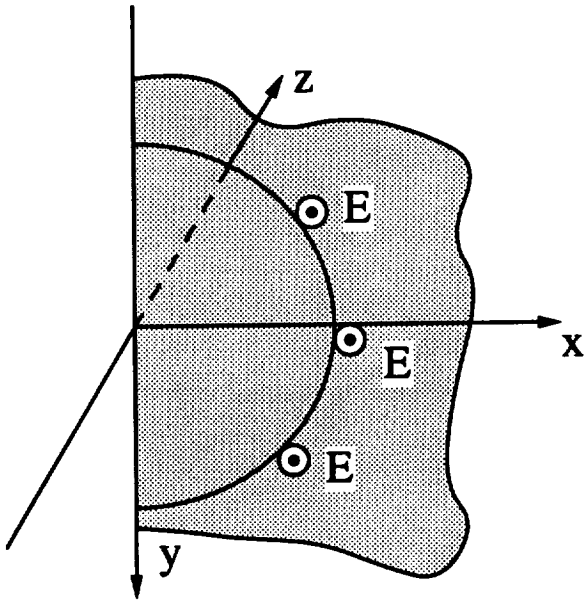


Figure 4.2: H-Plane of the LTSA

planes (E and H). Therefore, E and H plane data are measured by considering Figure 4.1 and Figure 4.2, and measuring the field component in the direction shown by the direction of E in the mentioned figures. In the E-Plane, $\phi = 0$ degrees and θ is varying, whereas in the H-Plane, $\theta = 90$ degrees and ϕ is varying. These can be better understood by considering Figure 4.3 which shows the LTSA and the standard gain antenna positioning. In Figure 4.3 it is assumed that the test antenna (LTSA) is used in receive mode and the polarization of the transmit antenna is as shown, however, everything remains for the transmit mode operation of the LTSA. Considering Figure 4.3, co-polar measurements can be listed as:

- E-Plane: $\beta_r = 90$ degrees, $\beta_t = 90$ degrees.
- H-Plane: $\beta_r = 0$ degrees, $\beta_t = 0$ degrees.
- D-Plane: $\beta_r = 45$ degrees, $\beta_t = 45$ degrees.

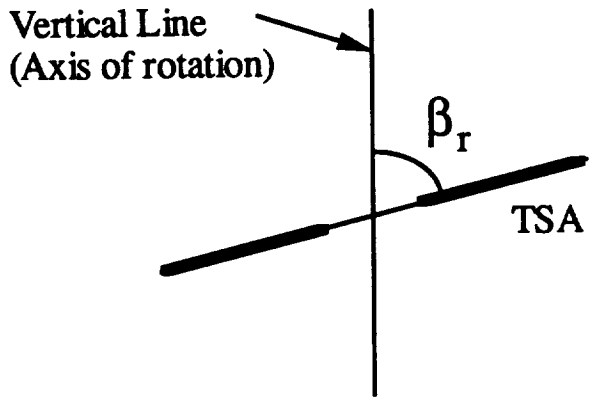
For cross-polar measurements, changing the polarization of the transmit antenna will be sufficient, which means changing β_t. This measurement strategy for the cross-polarized fields conforms to the third definition of Ludwig [56]. Therefore, cross-polar measurements can be done with the following set-up.

- E-Plane: $\beta_r = 90$ degrees, $\beta_t = 0$ degrees.
- H-Plane: $\beta_r = 0$ degrees, $\beta_t = 90$ degrees.

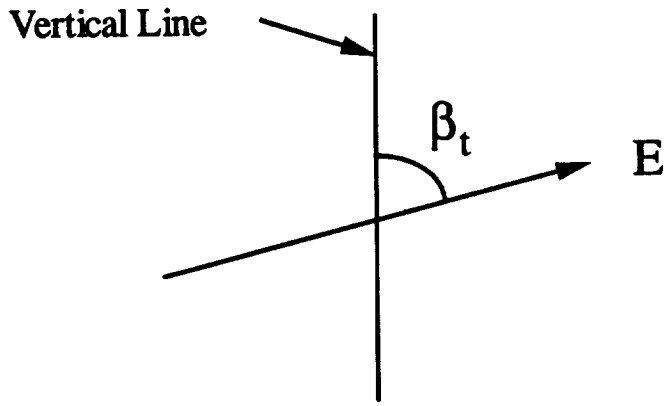
- D-Plane: $\beta_r = 45$ degrees, $\beta_t = -45$ degrees.

4.2 Verification of Computed Results

Numerical results obtained using the code are tested and verified both computationally and experimentally. First, the unequal-size rectangular sectioning scheme of section 3.2 is tested. In order to do this, the analysis is extended to handle all four edges of the skew-plate antenna of Figure 1.2. The results of this analysis is compared to those obtained from another code which uses skew segments, and hence models the skew-plate antenna exactly in the geometrical sense. In many cases that are computed, very good agreement is observed between the results. Figures 4.4 and 4.5 show the comparison for the E-Plane and H-Plane co-polar radiation patterns, respectively, for a skew-plate antenna with $L = \lambda_0$, $H = 0.5\lambda_0$, $\alpha = 5\,\text{deg}$ and $W_f = 0.004\lambda_0$. In the skew segmentation model, 7 segments across the length and 4 segments across the height are used. In the rectangular model, the number of divisions are 8 for the length and 5 for the height. As can be seen from Figures 4.4 and 4.5, the two computations agree very well, the largest difference between the two being about 1 dB. Considering that all four sides are approximated with unequal-size rectangular modeling and the fact that there is only one edge in the actual LTSA geometry, which is approximated in this manner, it is concluded that the accuracy would be even better in that case.



(a)



(b)

Figure 4.3: Antenna configurations a) LTSA positioning, b) Standard gain antenna positioning.

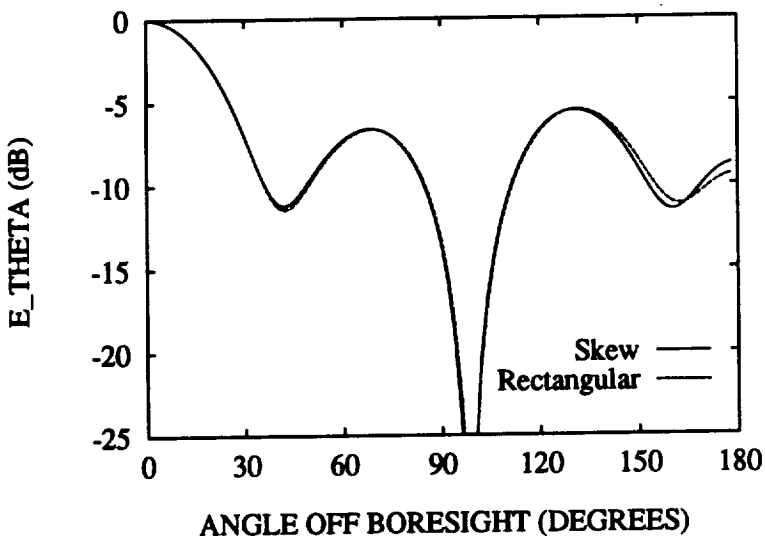


Figure 4.4: Comparison of E-Plane radiation patterns for skew-plate and unequal-size rectangular modeling ($L = 1.0\lambda_0$, $H = 0.5\lambda_0$, $W_f = 0.004\lambda_0$, $\alpha = 5$ degrees).

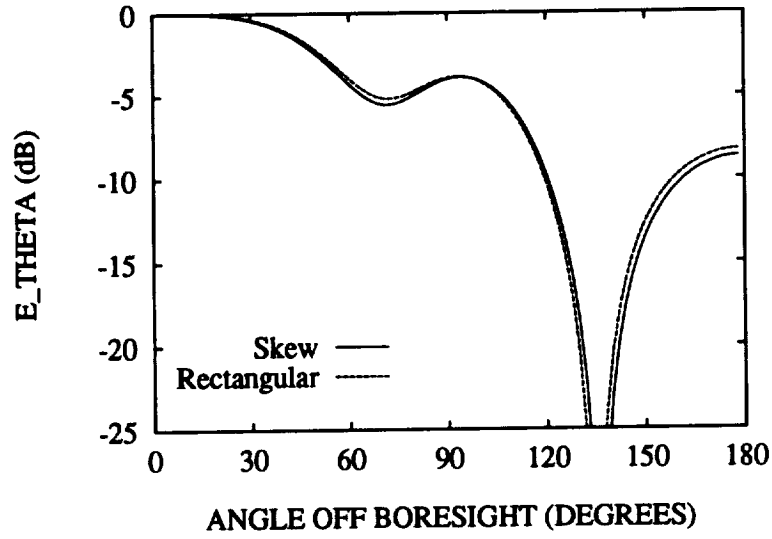


Figure 4.5: Comparison of H-Plane radiation patterns for skew-plate and unequal-size rectangular modeling ($L = 1.0\lambda_0$, $H = 0.5\lambda_0$, $W_f = 0.004\lambda_0$, $\alpha = 5$ degrees).

The convergence of the unequal size rectangular sectioning model is usually obtained using 6 to 7 segments per wavelength across the length and 4 to 5 segments per wavelength across the height of the LTSA. The number of segments required across the length is larger because the unequal-size rectangular sectioning is more sensitive to segmentation across the length. Figures 4.6 and 4.7 show the radiation pattern comparison for two different segmentations in the E and H planes, respectively, of a skew-plate antenna with parameters $L = 5.2\lambda_0$, $H = 0.9\lambda_0$, $W_f = 0.06\lambda_0$ and $\alpha = 7\,\text{deg}$. The data represented by solid lines in these figures are obtained by

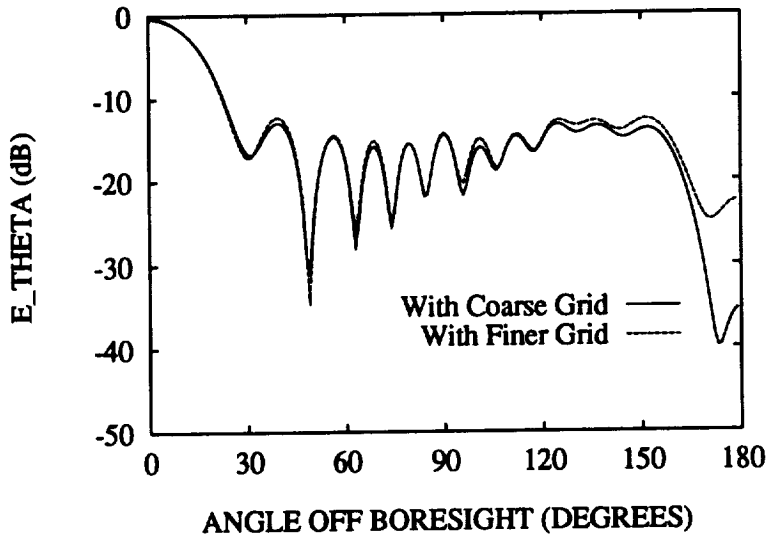


Figure 4.6: E-Plane radiation pattern for a skew-plate antenna with two different segmentation ($L = 5.2\lambda_0$, $H = 0.9\lambda_0$, $W_f = 0.06\lambda_0$, $\alpha = 7$ degrees).

using 30 segments across the length and 6 segments across the height, whereas the dotted lines are obtained using 35 and 7 segments across the length and the height,

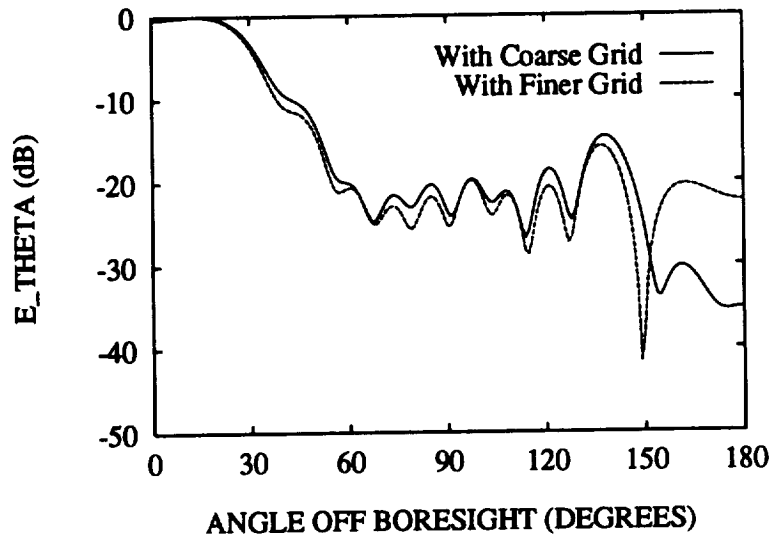


Figure 4.7: H-Plane radiation pattern for a skew-plate antenna with two different segmentation ($L = 5.2\lambda_0$, $H = 0.9\lambda_0$, $W_f = 0.06\lambda_0$, $\alpha = 7$ degrees).

respectively. As can be seen from the figures, the analysis results are very close to each other until 150 degrees. The effect of the difference in segmentation is observed only after 150 degrees, displaying the good convergence behavior of the algorithm. The parameters of this antenna are chosen the same as that analyzed in [10, 5]. The results of Figures 4.6 and 4.7 agree very well with those reported in [10].

The ultimate test on any electromagnetic modeling code is done by calculating the near fields at the conducting boundary and in the dielectric region and checking the calculations for the satisfaction of the boundary conditions [17]. The cost of this test is the same as the solution of the MoM matrix equation and hence is not practical for large problem sizes. However, an easier approach to test the near-field behavior of a code is to calculate the current distribution on/in the structure and check it for

abnormalities in the amplitude and phase. This approach is used in this work to test the near-field performance of the code. Figures 4.8 to 4.11 show the magnitude and phase plots of the current on a LTSA with $L = 5.2\lambda_0$, $H = 1.5\lambda_0$, $W_f = 0.06\lambda_0$ and $\alpha = 7\,\text{deg}$.

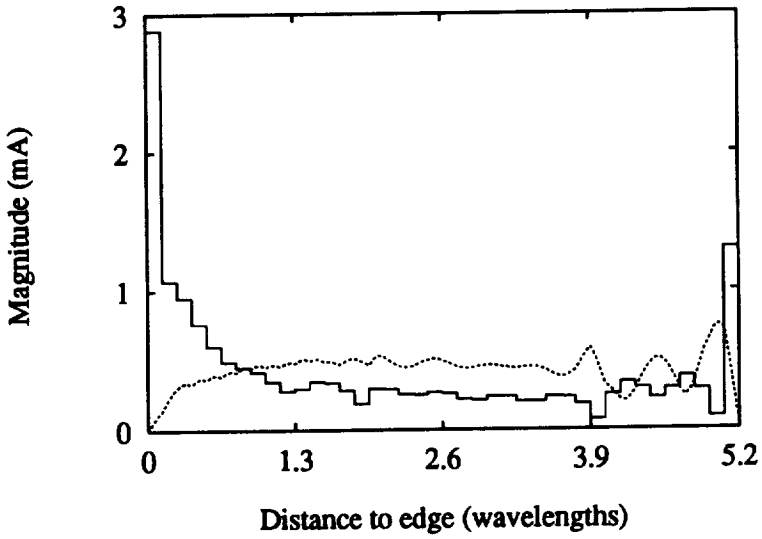


Figure 4.8: Magnitude of antenna current along $z = 0.75\lambda_0$. — : J_z, - - - : J_x.

Figures 4.8 and 4.9 show the magnitude and phase variation of the x and z components of the antenna current along a horizontal (z) cut at $0.75\lambda_0$ away from the lower antenna edge. Figures 4.10 and 4.11 gives the same components for horizontal (x) cut at $2.53\lambda_0$ away from the antenna edge. The traveling wave nature of the z component of the current is evident in Figures 4.8 and 4.9, and both the amplitude and the phase are free of abnormal behavior. The current displays a standing wave

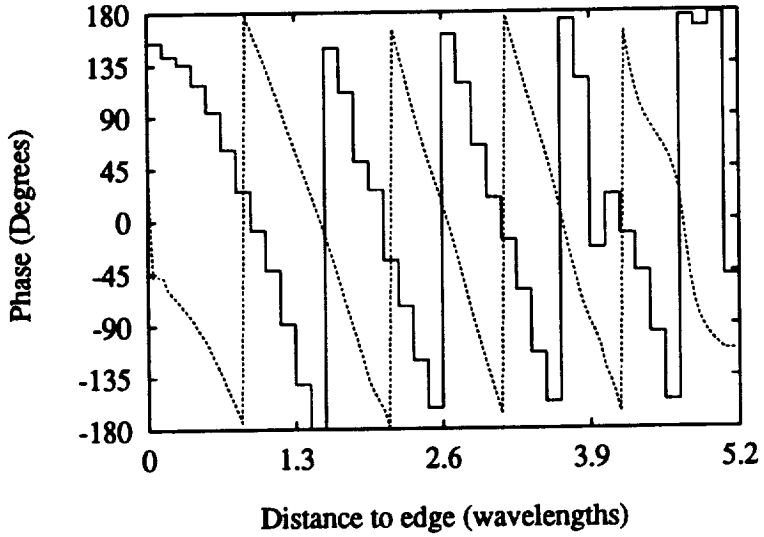


Figure 4.9: Phase of antenna current along $z = 0.75\lambda_0$. — : J_z, - - - : J_x.

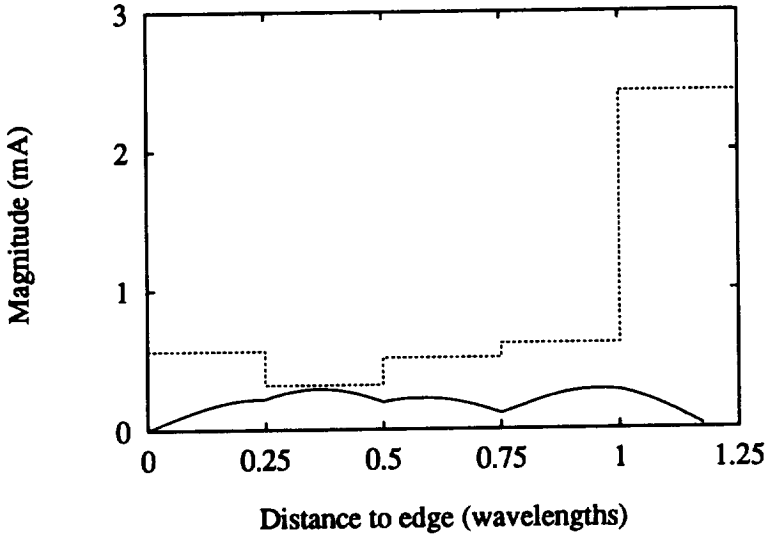


Figure 4.10: Magnitude of antenna current along $x = 2.53\lambda_0$. — : J_z, - - - : J_x.

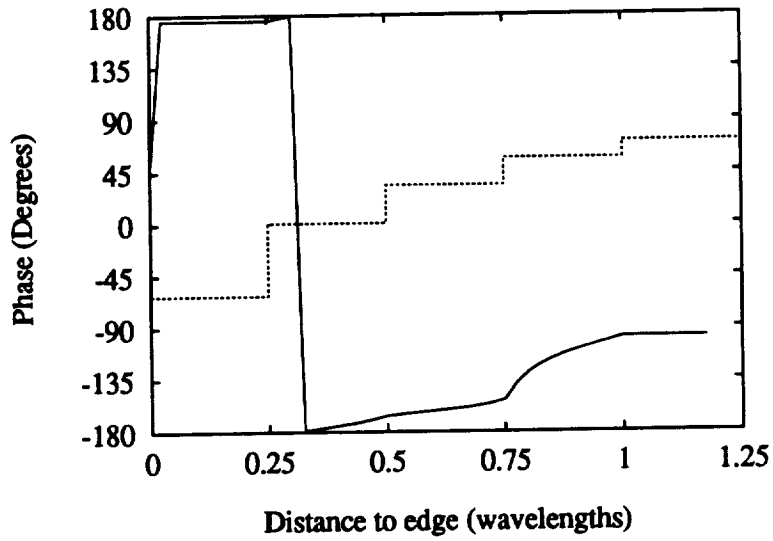


Figure 4.11: Phase of antenna current along $x = 2.53\lambda_0$. — : J_z, - - - : J_x.

nature in z direction, as shown in Figures 4.10 and 4.11. This should be expected because the antenna height is small ($1.5\lambda_0$) and the current bounces back and forth between the two edges of the antenna.

Computed results are also verified by experimentation. For this purpose, two antennas are built and measurements are taken in E, H and D planes. The first antenna is intended to check the air LTSA results and was built using 5-mil brass sheet and supported using styrofoam which has a permittivity (1.05), very close to that of free-space. Microstripline to slotline transition is used in the feeding section of the antenna which extends $0.5\lambda_0$, where λ_0 is the wavelength at the operating frequency of 9 GHz. The feeding part of the antenna is designed using 31-mil, $\epsilon_r = 2.33$, Duroid substrate. The substrate is terminated abruptly at the apex of the antenna, where

the taper starts. The guidelines given in [50] are used to design the microstripline to slotline transition which resulted in slotline impedance of 138.2Ω and microstripline impedance of 120Ω. This slotline impedance is achieved with $W_f = 0.659$ mm ($0.01977\lambda_0$ at 9 GHz). The wavelength in the slotline is 2.88 cm ($0.864\lambda_0$), whereas the microstripline wavelength is 2.4793 cm ($0.74379\lambda_0$). The width of the microstripline for 120Ω characteristic impedance is found as 0.4171 mm ($0.01251\lambda_0$). In order to match the microstripline to the 50Ω output impedance of the test equipment, a quarter wave impedance transformer is designed at the center frequency of 9 GHz. The final design is shown in Figure 4.12, where $H = 1.5\lambda_0$, $L_i = 0.5\lambda_0$, $xs0 = 0.216\lambda_0$, $c = 0.01251\lambda_0$, $d = 0.03392\lambda_0$, $e = 0.0679\lambda_0$, $f = 0.63221\lambda_0$, and $g = 0.45\lambda_0$.

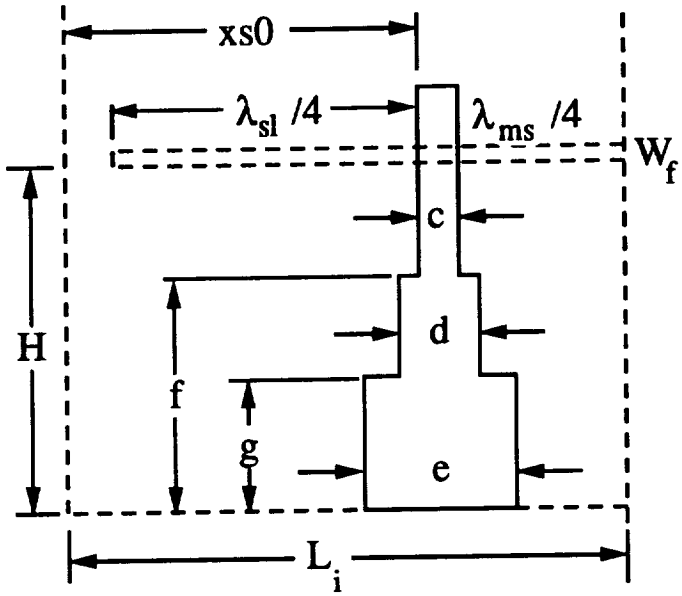


Figure 4.12: Feed design of the test antenna

The comparison between computation and measurement for co-polar E and H plane radiation patterns is given in Figure 4.13. Figure 4.14 shows the comparison for the co-polar and cross-polar radiation patterns in D plane. In the computation the dielectric support at the feeding part of the antenna is modeled rigorously. The microstripline feed is modeled using only 120Ω characteristic impedance line extending to the edge of the antenna. In the numerical model 48 segments across the length and 6 segments across the height of the conductor are used. The microstripline is modeled by 17 segments, resulting in 1059 conductor unknowns. The dielectric segments across length, width, and height are 4, 34 and 1, respectively, which give 408 dielectric current unknowns. The solution time for this case was 2062 CPU seconds on CRAY Y-MP.

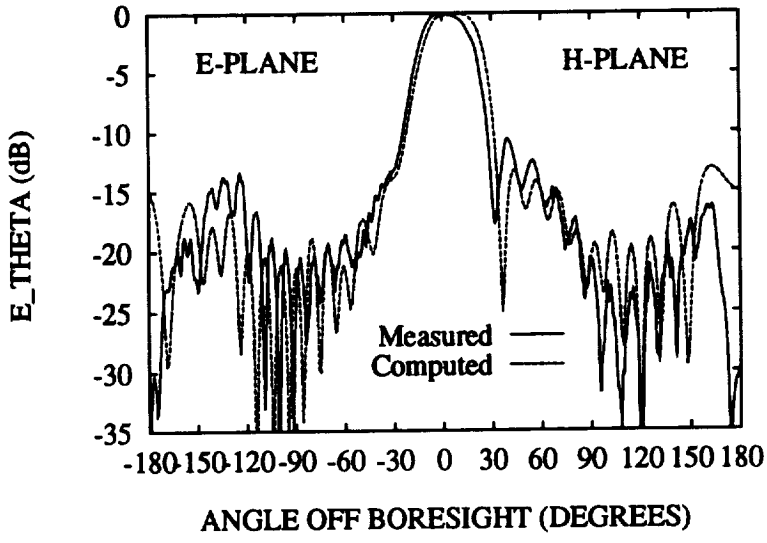


Figure 4.13: Measured and computed co-polar radiation patterns for LTSA in air ($L = 5.5\lambda_0$, $L_i = 0.5\lambda_0$, $H = 1.5\lambda_0$, $W_f = 0.02\lambda_0$, $\alpha = 7$ degrees).

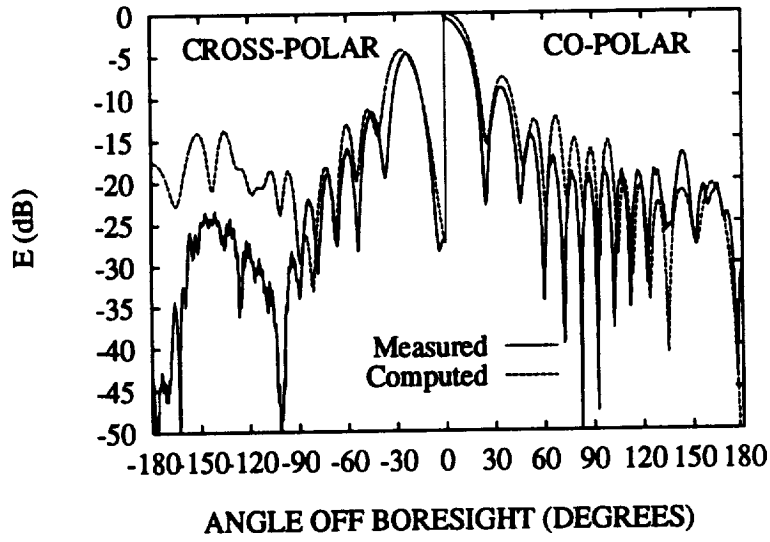


Figure 4.14: Measured and computed co-polar and cross polar D-Plane radiation pattern for LTSA in air ($L = 5.5\lambda_0$, $L_i = 0.5\lambda_0$, $H = 1.5\lambda_0$, $W_f = 0.02\lambda_0$, $\alpha = 7$ degrees).

As can be seen from Figures 4.13 and 4.14, quite good agreement is obtained between the computed results and measured data. The computed pattern predicts the main beam and the first side lobe level correctly. The pattern shapes also agree well. Slight discrepancies between the two is thought to be resulting from the alignment errors during the manufacturing of the test antenna and from the effect of the adhesive used to attach the antenna to the styrofoam. The difference between the cross-polar measured and calculated data below -90 degrees results from the use of an absorber piece over the source region during the measurements. However, the maximum cross polarization level and the cross polarized pattern is predicted correctly by the code until this angle. The effect of the absorber is negligible for the co-polar measurements,

which leads to the good agreement for this part of the comparison.

The second test antenna is built using a 31-mil thick, $\epsilon_r = 2.33$ Duroid substrate, and is used to evaluate the dielectric LTSA calculations. This antenna has the same feed design values as the air LTSA test antenna. Figures 4.15 and 4.16 show the comparison for the E and H plane co-polar radiation patterns and co-polar and cross-polar radiation patterns for the D plane , respectively. The computed patterns for this case is obtained using 36 segments in length and 6 segments in height for the conductor parts, 17 segments for the microstripline, 40 segments in length, 1 segment in height and 24 segments in width for the dielectric region. The total number of unknowns is 3541 of which 661 is the conductor unknowns. The solution time for this case was 2766 CPU seconds on CRAY Y-MP.

A very good agreement is observed between the computed and measured data for this case. The code predicts the shape and the amplitudes of the radiation patterns accurately for this antenna as well. Actually, the agreement is better for this antenna since the dielectric support of the antenna extends through the whole length of the antenna. In the air case test antenna, the dielectric support is terminated in the feed section and hence the diffraction from the dielectric edge can be appreciable and is not handled by the code due to choice of the basis functions in the dielectric. Also, the slight discrepancies after 150 degrees in dielectric LTSA comparisons is again attributed to the use of an absorber block in the measurements over the input section of the antenna which is not modeled by the code.

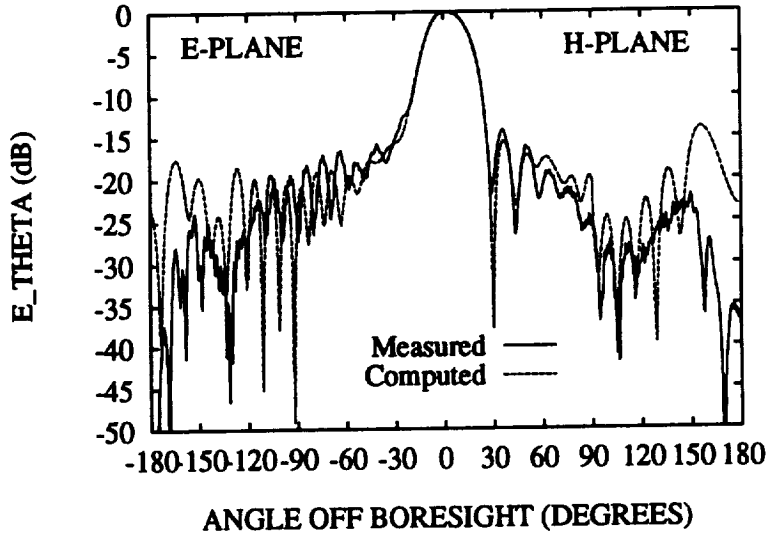


Figure 4.15: Measured and computed co-polar E and H-Plane radiation patterns for a dielectric LTSA ($\epsilon_r = 2.33$, $d = 0.02362\lambda_0$, $L = 5.5\lambda_0$, $L_i = 0.5\lambda_0$, $H = 1.5\lambda_0$, $W_f = 0.02\lambda_0$, $\alpha = 7$ degrees).

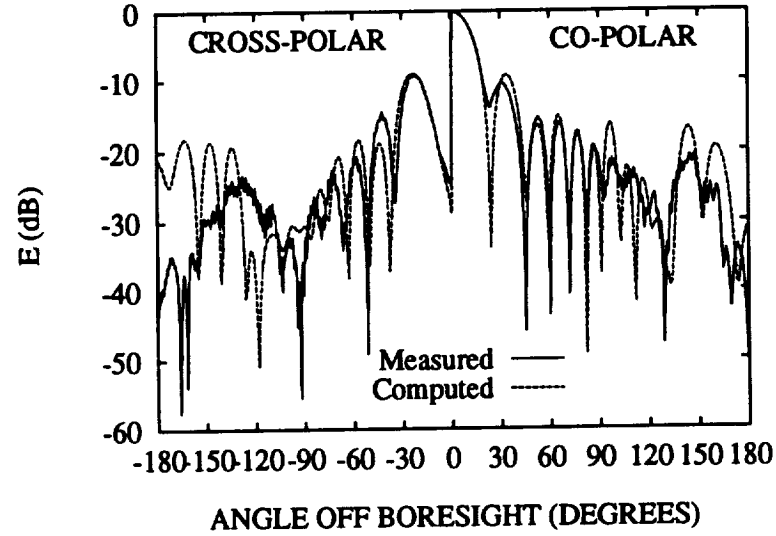


Figure 4.16: Measured and computed co-polar and cross-polar D-Plane radiation patterns for a dielectric LTSA ($\epsilon_r = 2.33$, $d = 0.02362\lambda_0$, $L = 5.5\lambda_0$, $L_i = 0.5\lambda_0$, $H = 1.5\lambda_0$, $W_f = 0.02\lambda_0$, $\alpha = 7$ degrees).

The comparisons with available data in the literature and with experimental data that is presented in this section leads to the conclusion that the theoretical model and the code can predict the radiation characteristics of air or dielectric linearly tapered slot antennas with reasonable accuracy.

4.3 Computed Results for Air Tapered Slot Antennas

In this section, computed results for air LTSA's will be presented. In [4], Janaswamy has observed that as the antenna height is decreased for fixed length and taper angle, better radiation patterns are achievable. In order to address this question, a parametric study of air LTSA's is planned and carried out. Since the behavior of the dielectric LTSA is quite similar to the air LTSA, conclusions drawn for the air LTSA can also be applied to the dielectric LTSA. In the parametric study, the apex width, W_f, of the LTSA is chosen as 2 mm (0.06λ at 9 GHz) and the antennas are assumed to be in the receiving mode with a diode soldered at the apex. The diode is modeled by a strip dipole of width 0.02λ and length 0.2λ. Three levels for each of the parameters L, H and α are chosen. The levels for α are 5 deg, 7 deg and 9 deg. H assumes the values of λ, 1.5λ and 2λ, whereas L varies as λ, 3λ and 5λ. The analysis is valid for any frequency provided that all dimensions are the same used in the analysis in terms of the wavelength. These three levels for α, H and L resulted in 27 numerical experiments. Co-polar radiation patterns in E, H and D planes and

cross-polar radiation patterns in D plane are computed for all experiments. Table 4.1 shows the experiments and the corresponding calculated figures of merit. The figures of merit used in Table 4.1 are:

1. EBW: 3-dB beamwidth in the co-polar E-Plane radiation pattern,
2. ESL: First sidelobe level in the E-Plane radiation pattern,
3. HBW: 3-dB beamwidth in the co-polar H-Plane radiation pattern,
4. HSL: First sidelobe level in the H-Plane radiation pattern,
5. DBW: 3-dB beamwidth in the co-polar D-Plane radiation pattern,
6. DSL: First sidelobe level in the D-Plane radiation pattern,
7. DXL: Peak cross-polarization level in the D-Plane radiation pattern.

Since the antenna and the receiving strip dipole are symmetric about the plane $z = 0$, the cross polarization is theoretically zero $(-\infty dB)$, in the E and H planes of the antenna. This fact is also verified in the calculations. Considering Table 4.1, the following observations are made:

- As H decreases for a fixed L and α, EBW first decreases and then starts to increase again. ESL also behaves in the same manner. However, HBW increases steadily whereas HSL decreases. The D-Plane radiation pattern follows the same trend as the E-Plane with first decreasing then increasing DBW and

Table 4.1: Results for the air LTSA study

Exp. No	L	H	α	EBW	ESL	HBW	HSL	DBW	DSL	DXL
1	5	2	9	32.0	-11.41	41.2	-5.81	38.7	-7.66	-2.98
2	5	2	7	34.84	-14.02	41.0	-6.06	40.44	-6.72	-2.28
3	5	2	5	38.72	-15.08	42.0	-6.49	42.58	-5.84	-1.37
4	5	1.5	9	18.6	-14.05	48.4	-10.07	24.86	-9.76	-6.54
5	5	1.5	7	20.0	-14.97	48.36	-10.35	26.28	-8.66	-5.57
6	5	1.5	5	20.84	-12.20	48.52	-10.65	27.2	-7.60	-4.95
7	5	1.0	9	32.9	-11.52	65.0	-20.38	36.24	-6.01	-4.18
8	5	1.0	7	34.5	-15.62	61.84	-19.64	36.36	-5.29	-3.91
9	5	1.0	5	40.9	-13.05	64.64	-20.07	38.96	-4.86	-3.28
* 10	3	2	9	43.2	-12.50	52.2	-5.63	52.1	-7.52	-2.38
* 11	3	2	7	44.0	-11.4	53.12	-5.62	52.40	-6.82	-2.00
* 12	3	2	5	45.0	-9.26	54.2	-5.45	52.62	-6.51	-1.86
13	3	1.5	9	26.0	-4.31	59.24	-6.64	35.74	-5.93	-2.97
14	3	1.5	7	26.5	-4.10	53.96	-7.19	34.1	-6.26	-3.38
15	3	1.5	5	25.12	-3.79	54.0	-7.21	33.0	-5.77	-3.04
16	3	1.0	9	38.2	-12.31	71.22	-17.47	44.2	-5.88	-4.72
17	3	1.0	7	41.6	-10.02	72.2	-18.42	46.1	-4.75	-4.17
18	3	1.0	5	45.3	-7.7	73.4	-17.31	48.0	-3.93	-3.73
** 19	1	2.0	9	101.4	-2.87	87.2	-6.48	72.0	-3.94	0
** 20	1	2.0	7	100.6	-2.48	86.0	-5.79	71.4	-3.67	0
** 21	1	2.0	5	100.0	-2.08	84.6	-5.05	71.0	-3.38	0
** 22	1	1.5	9	95.6	-2.45	98.5	-3.78	94.0	-3.21	0
** 23	1	1.5	7	95.0	-1.99	92.4	-3.20	93.0	-2.92	0
** 24	1	1.5	5	94.0	-1.54	87.0	-2.56	91.6	-2.63	0
** 25	1	1.0	9	81.5	-2.49	80.4	-3.66	96.2	-5.13	0
** 26	1	1.0	7	81.0	-2.03	79.5	-3.25	95.6	-4.76	0
** 27	1	1.0	5	80.4	-1.55	78.8	-2.78	93.8	-4.37	0

DSL. The peak cross polarization level in the D-Plane behaves differently for antennas of different length. For $L = 5\lambda$, DXL first decreases then starts to increase, however, for $L \leq 3\lambda$ it steadily decreases as H decreases. Figures 4.17 and 4.18 display these behaviors. The experiment numbers of Table 4.1 are

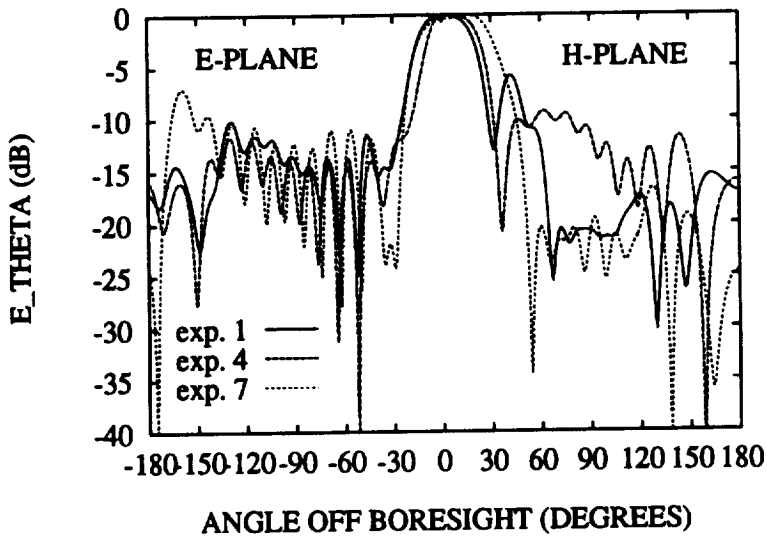


Figure 4.17: Variation of E and H-Plane patterns of LTSA's with H.

used to identify the data in these figures.

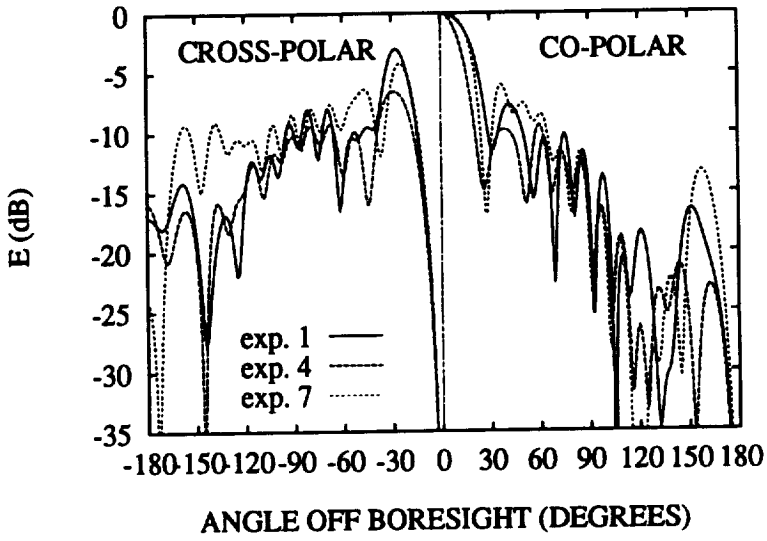


Figure 4.18: Variation of D-Plane pattern of LTSA's with H.

- As α decreases, EBW increases. ESL decreases slightly for $L = 5\lambda$. For shorter antennas, EBW remains nearly the same, however ESL increases. The H-Plane of the antenna is not as sensitive to variation in α, the main beam and sidelobe levels and the shape remain nearly the same, while lobe locations change slightly. Only for $H = \lambda$ and $L = 5\lambda$, a slight decrease of HBW is observed with decreasing α. DBW, DSL and DXL increase with decreasing α, the largest deviation in DXL being for large L. These variations are shown in Figures 4.19 and 4.20.

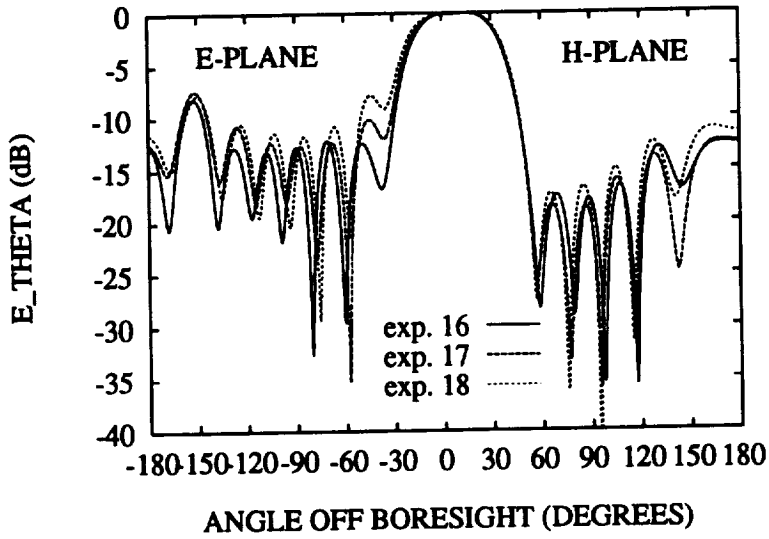


Figure 4.19: Variation of E and H-Plane patterns of LTSA's with α.

- As L increases, the antenna behaves as expected. All of the 3 dB beamwidths decrease, with decreasing sidelobe levels and peak cross polarization level in D-

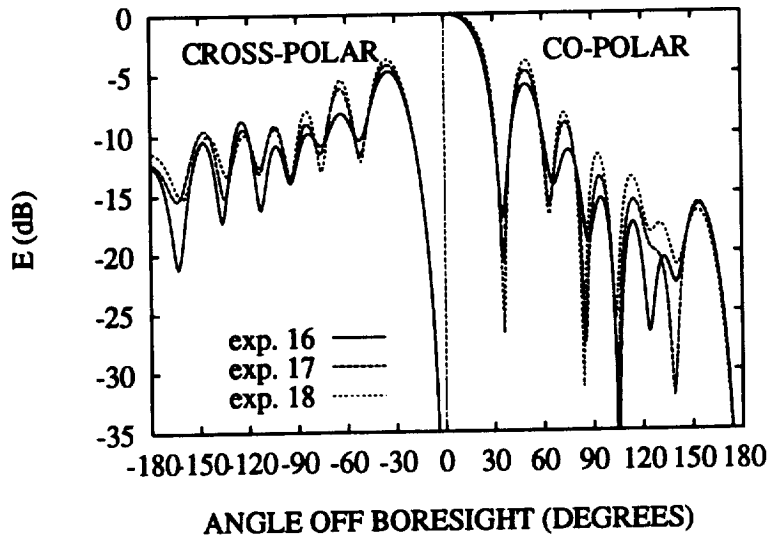


Figure 4.20: Variation of D-Plane pattern of LTSA's with α.

Plane. However, an interesting behavior is observed for short antennas when the total height of the antenna is larger than the length. In these cases, the maximum in the E-Plane radiation pattern is not obtained in the boresight direction. These cases are marked with * in Table 4.1. For short antennas, the current does not have the traveling wave nature in x-direction any more. When individual segment currents in x and z directions are considered for the LTSA geometry, a similarity to the skewed linear antenna can be a possible explanation for this behavior. Depending on the included angle, the skewed line antenna can create a radiation pattern which has a maximum at a direction other than boresight. When the length of the antenna is further reduced, maxima of the computed patterns are obtained in the D Plane and in the

cross polarized direction (cases 19 to 27). This observation is again attributed to the fact that the radiation due to z directed currents are more important than x directed currents. These cases are marked with ** in Table 4.1. The behavior of the antenna as a function of L is demonstrated in Figures 4.21 and 4.22.

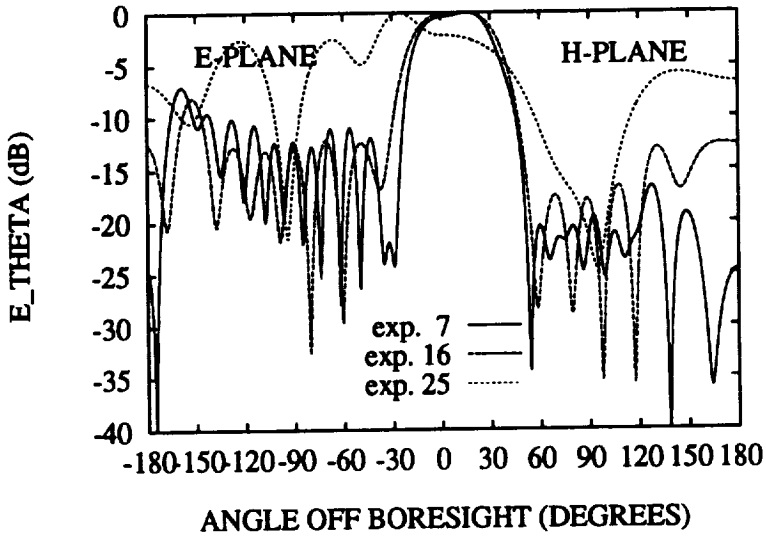


Figure 4.21: Variation of E and H-Plane patterns of LTSA's with L.

In general, it is observed that the peak cross polarization level of the antenna is quite high for the cases considered. However, it is interesting to note that a better radiation pattern can be obtained by decreasing the antenna height for a fixed L and α. Another interesting observation is that somewhat better antenna characteristics can still be obtained for short antennas ($L = \lambda$, for example) by keeping the antenna

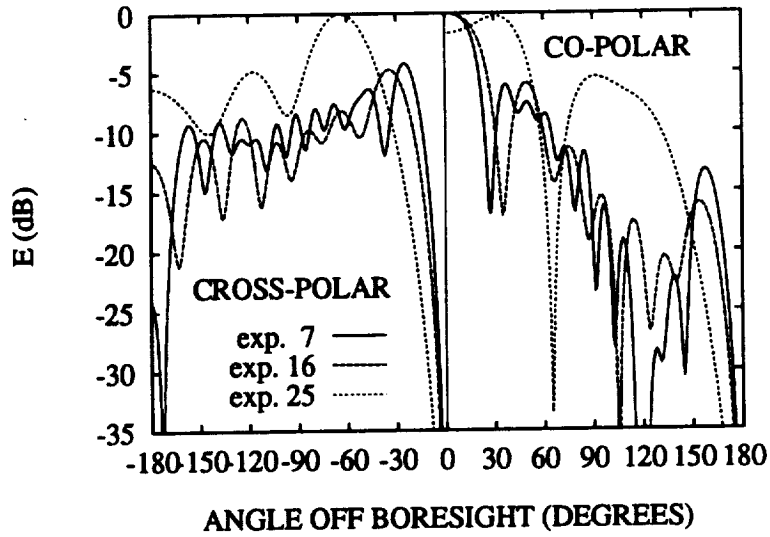


Figure 4.22: Variation of D-Plane pattern of LTSA's with L.

height small as well. This way, although the sidelobe levels and the cross polarization level are not as small as one can obtain from a longer antenna, the maximum is still attained in the boresight direction in the E-Plane. For array applications, this might be a useful design criterion since the sidelobe and cross polarization levels and beamwidth heavily depend on the array factor of the structure as well.

4.4 Computed Results for Dielectric Tapered Slot Antennas

In this section, sample results for dielectric LTSA's will be given. In order to investigate the effect of the dielectric permittivity, the same antenna geometry with a

receiving diode is computed with three different permittivities of the dielectric support. The results with the antenna parameters are given in Figures 4.23 and 4.24. It is seen that, as the permittivity increases the E and H-Plane pattern sidelobe

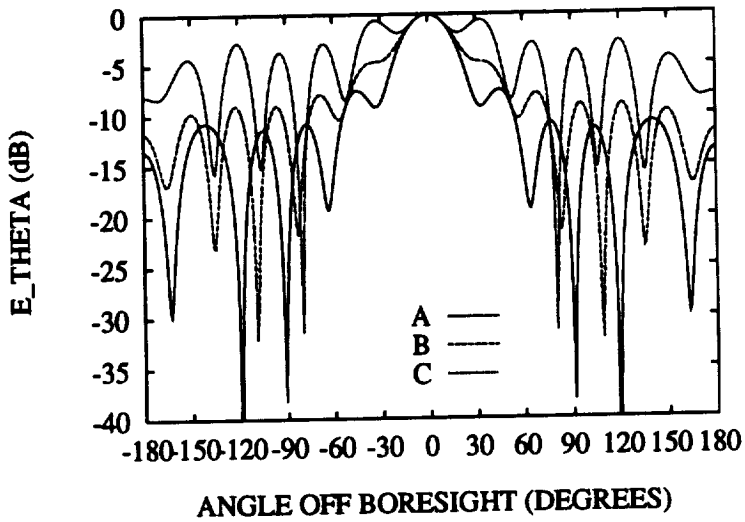


Figure 4.23: Variation of the E-Plane pattern for LTSA's with ϵ_r. A: $\epsilon_r = 2.33$, B: $\epsilon_r = 4.0$, C: $\epsilon_r = 5.0$ ($L = 2.0\lambda_0$, $H = 0.4\lambda_0$, $W_f = 0.01\lambda_0$, $d = 0.03\lambda_0$, $\alpha = 5$ degrees).

levels increase. The 3 dB beamwidth in the E-Plane remains essentially the same for this particular antenna geometry, whereas the H-Plane pattern beamwidth decreases. This should be expected since a higher percentage of the radiated power is trapped in the dielectric region of the antenna as the permittivity increases. Also, with increasing permittivity, the H-Plane pattern becomes more asymmetrical.

The analysis of the antenna of Figure 4.23 with $\epsilon_r = 2.33$ with changing dielectric thickness is shown in Figures 4.25 and 4.26. The same kind of behavior is observed

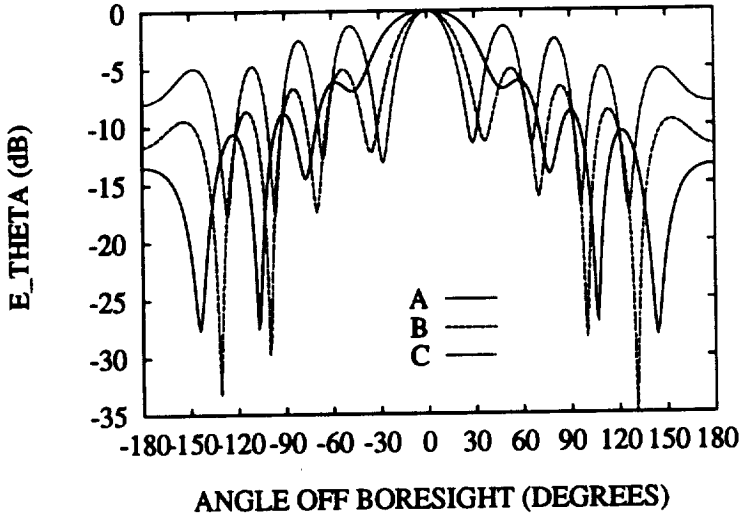


Figure 4.24: Variation of the H-Plane pattern for LTSA's with ϵ_r. A:$\epsilon_r = 2.33$, B:$\epsilon_r = 4.0$, C:$\epsilon_r = 5.0$ ($L = 2.0\lambda_0$, $H = 0.4\lambda_0$, $W_f = 0.01\lambda_0$, $d = 0.03\lambda_0$, $\alpha = 5$ degrees).

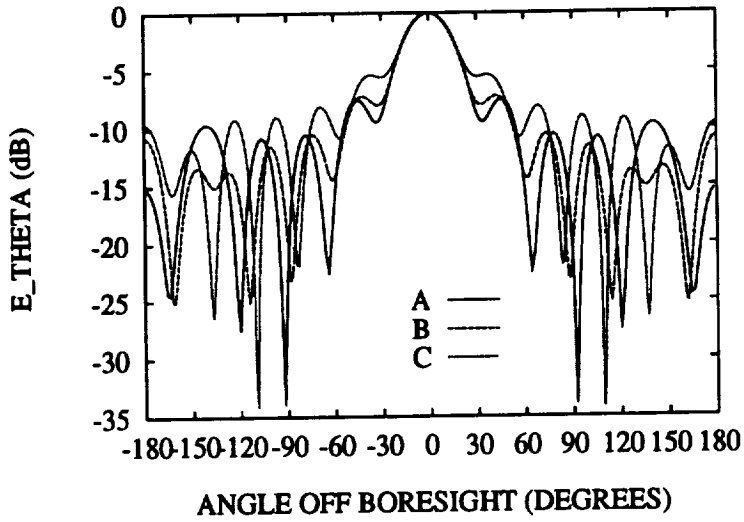


Figure 4.25: Variation of the E-Plane pattern for LTSA's with dielectric thickness, d. A: $d = 0.02\lambda_0$, B: $d = 0.06\lambda_0$, C: $d = 0.1\lambda_0$ ($\epsilon_r = 2.33$, $L = 2.0\lambda_0$, $H = 0.4\lambda_0$, $W_f = 0.01\lambda_0$, $\alpha = 5$ degrees).

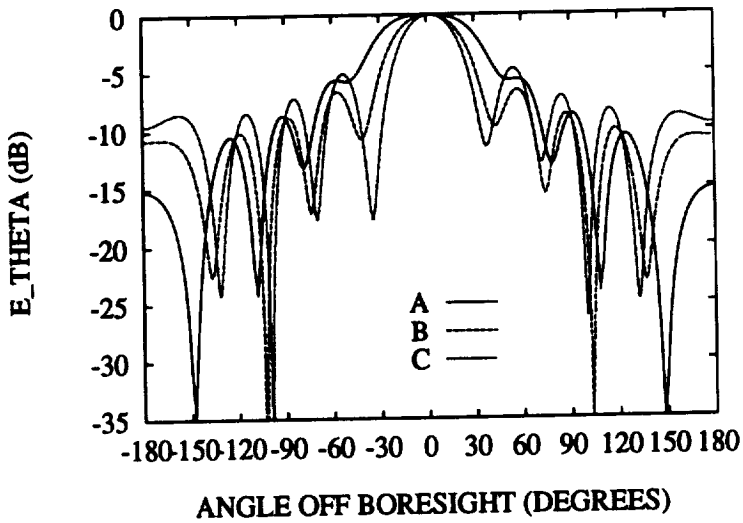


Figure 4.26: Variation of the H-Plane pattern for LTSA's with dielectric thickness, d. A: $d = 0.02\lambda_0$, B: $d = 0.06\lambda_0$, C: $d = 0.1\lambda_0$ ($\epsilon_r = 2.33$, $L = 2.0\lambda_0$, $H = 0.4\lambda_0$, $W_f = 0.01\lambda_0$, $\alpha = 5$ degrees).

with increasing dielectric thickness as with the increasing permittivity. However, in this case the variation in the sidelobe levels is not so large, a fact resulting from the small value of ϵ_r. To demonstrate this effect, a high permittivity antenna ($\epsilon_r = 9.8$) with changing dielectric thickness is analyzed and the results are shown in Figures 4.27 and 4.28. In this case, the effects are much more pronounced than the low permittivity case of Figures 4.25 and 4.26. These observations follow those reported in [4].

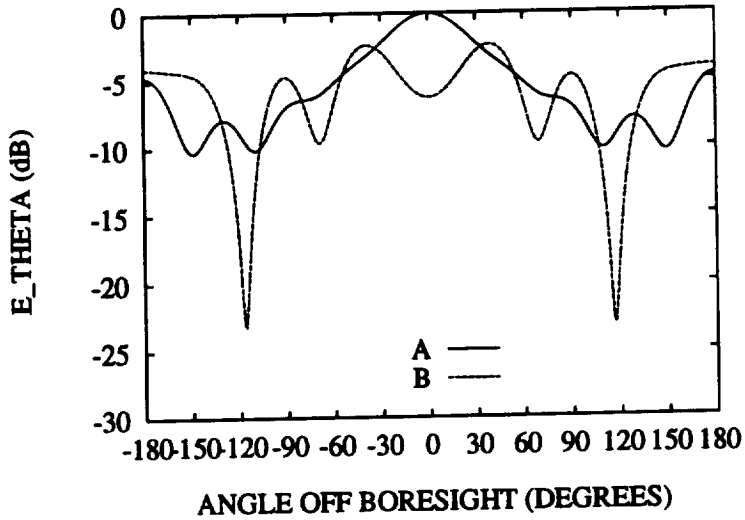


Figure 4.27: Variation of the E-Plane pattern for LTSA's with dielectric thickness, d, high e_r case. A: $d = 0.02\lambda_0$, B: $d = 0.04\lambda_0$ ($\epsilon_r = 9.8$, $L = 1.05\lambda_0$, $H = 0.38\lambda_0$, $W_f = 0.004\lambda_0$, $\alpha = 5.7$ degrees).

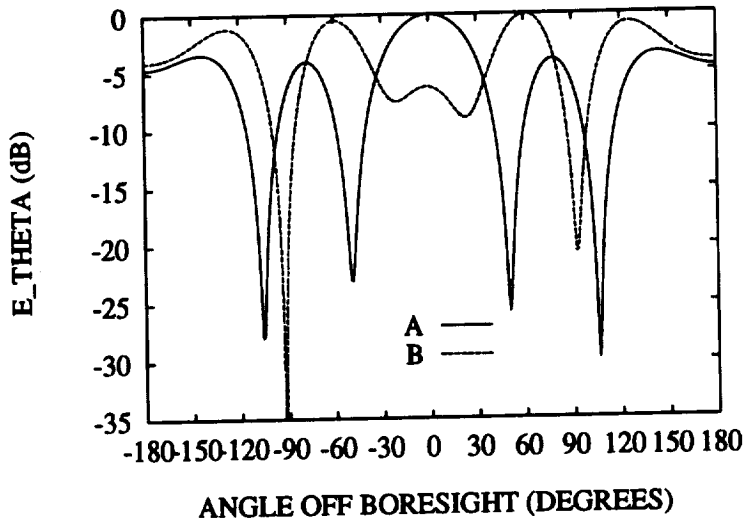


Figure 4.28: Variation of the H-Plane pattern for LTSA's with dielectric thickness, d, high e_r case. A: $d = 0.02\lambda_0$, B: $d = 0.04\lambda_0$ ($\epsilon_r = 9.8$, $L = 1.05\lambda_0$, $H = 0.38\lambda_0$, $W_f = 0.004\lambda_0$, $\alpha = 5.7$ degrees).

CHAPTER 5

COMPUTER CODE AND PERFORMANCE

5.1 Code

The block diagram of the code is shown in Figure 5.1. In the main program, **struc**, the geometry of the antenna is entered and the type of feeding is chosen. Subroutine **mom** calls the impedance matrix filling subroutines **filcc**, **filcd**, **fildc** and **fildd**. These calculate the conductor-conductor, conductor-dielectric, dielectric-conductor and dielectric-dielectric interactions respectively. **filvlt** calculates the right hand side vector of the MoM matrix equation (2.22). The matrix equation is solved by the **cgrad** routine which utilizes the conjugate gradient method of Chapter 3. Organization of the input and output files of the code and the listings of the routines can be found in the appendix (under separate cover).

5.2 CPU Time and Memory Requirements

Since large matrices result in the analysis, the performance of the code is optimized by both approximations in the calculations and by vectorization. The final version of the

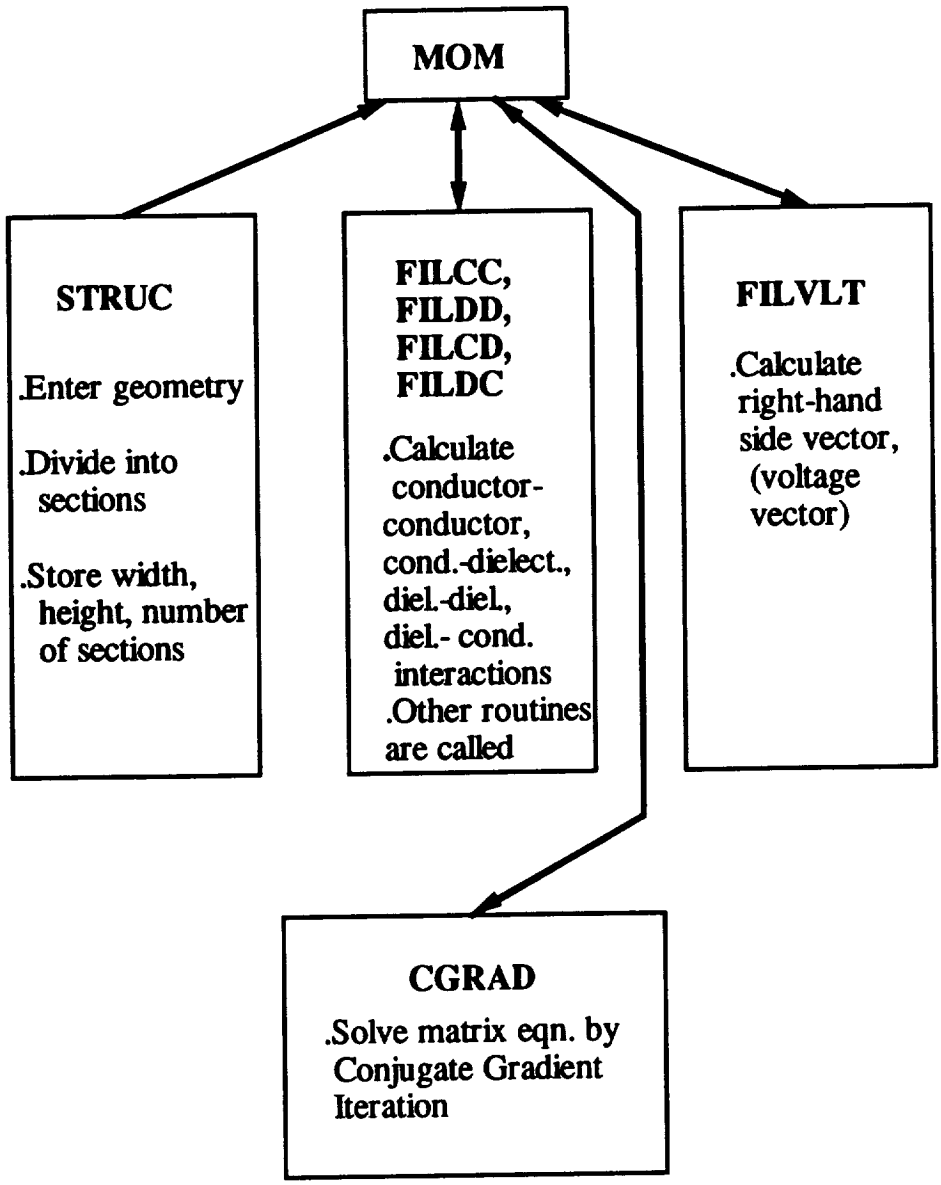


Figure 5.1: Block Diagram of the Code

code is adapted to and run on CRAY Y-MP of the North Carolina Supercomputing Center. The cost analysis of the code is carried out in order to estimate the necessary run times. For N conductor current unknowns and M dielectric polarization current

density unknowns, the matrix filling cost (FC) is given by

$$FC = O(N)n + O(NM)n + O(NM)nm + O(M)s \quad \text{Flops} \tag{5.1}$$

where n is the number of integration points, m is the number of subdivisions used in the computation of the dielectric-conductor interaction submatrix and s is the number of divisions used in the dielectric to dielectric interaction approximations. It is seen from (5.1) that FC is linearly proportional to $(N + M)$, the total number of unknowns.

The solution cost (SC) is given a

$$SC \leq \frac{(N+M)^3}{3} \text{ Flops} . \tag{5.2}$$

As mentioned earlier in Section 3.8, $(N + M)^3/3$ is the upper limit of SC. For most of the cases analyzed using the code, SC was much smaller than this limit because of the dominant diagonal of the resulting MoM matrix.

The performance of the code is monitored and enhanced throughout the work. Figure 5.2 shows the matrix fill time and the solution time of the code on Vector Alliant FX-40. The filling time increases linearly as predicted by (5.1), whereas the solution time increases faster, dominating the CPU time usage after about 400 unknowns. Figure 5.3 shows the comparison for the same cases analyzed using Alliant FX-40 and CRAY Y-MP. The big difference in the total run times in this figure results from the vectorization of the code and the high speed of the CRAY Y-MP

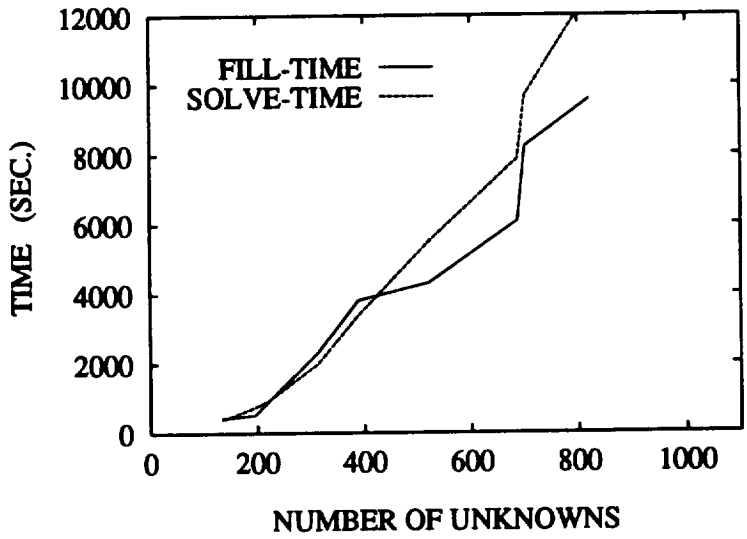


Figure 5.2: Matrix fill-time and solve time on Alliant FX-40 for air LTSA's

supercomputer. Another influencing factor is that the CRAY Y-MP is an actual memory machine, so that no time is lost for array reading and writing to and from the disk. Figure 5.4 shows the total run time as a function of number of unknowns for the dielectric LTSA's. All of these cases were calculated on the CRAY Y-MP because of the large CPU time that would be required otherwise. Here, it is worthwhile to note that the vectorization of the solution part of the code resulted in nearly linear behavior of the computation time instead of a higher power close to 3.

The limiting value of the number of unknowns in the method is set to be about $N + M = 5000$, where in a typical analysis $N = 1000$, and $M = 4000$. The run time memory requirement for this limit is approximately 26 MWords. All cases analyzed using the code resulted in less number of unknowns than 5000, and hence less mem-

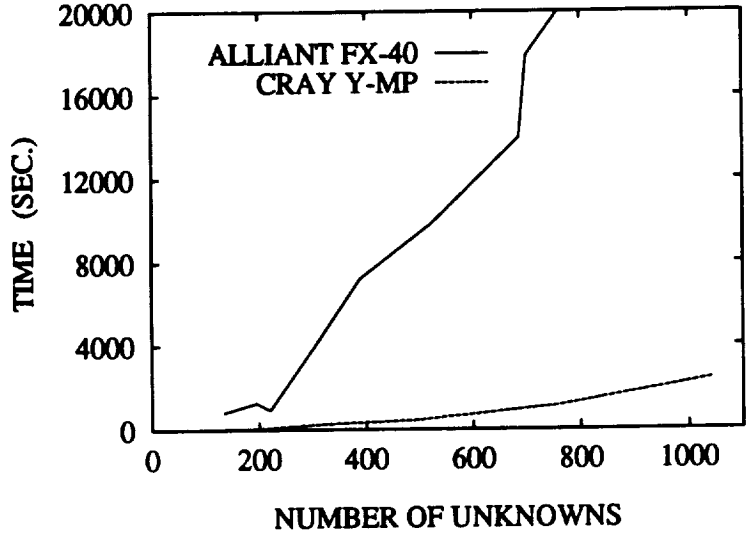


Figure 5.3: Total CPU time comparison for air LTSA's

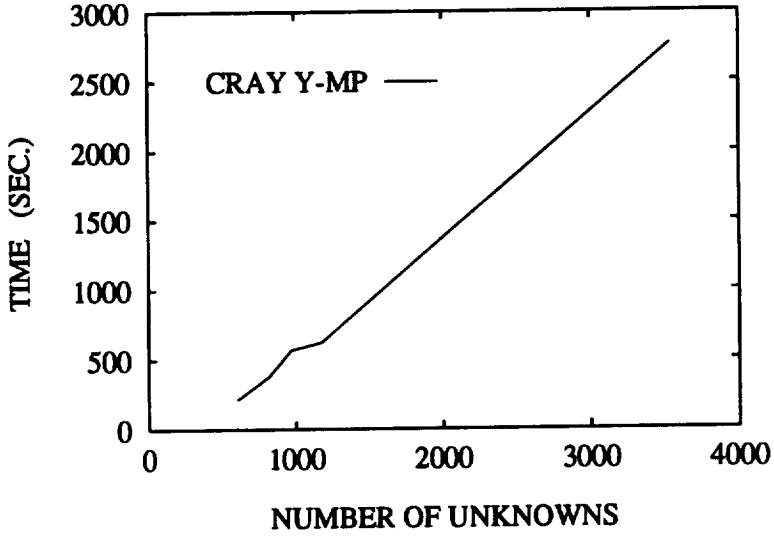


Figure 5.4: CPU time on CRAY Y-MP for dielectric LTSA's

ory requirement. The largest case that is analyzed using the code is the experimental dielectric LTSA of Chapter 4, which resulted in about 4000 unknowns. For higher permittivity dielectrics than that was used in the experiment, ($\epsilon_r = 2.33$), one would need more subdivisions in the dielectric. This, in turn, would reduce the solvable antenna dimensions. Therefore, as the permittivity increases, smaller antennas can be analyzed accurately with the code. Also, increasing the dielectric thickness would have the same effect since more segments than one would be needed across the thickness (y direction) of the dielectric. However, these statements are valid for current computational abilities and with the future developments in computer technology the solution of bigger problems with similar methods will be possible and less costly.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In this work, a Method of Moments model for the analysis of the Linearly Tapered Slot Antenna is developed. The conducting parts of the antenna, including the feed section, are approximated closely for the first time. The finite dielectric region is modeled rigorously by including the equivalent volume polarization current density as an unknown in the formulation. The use of Schelkunoff's equivalence principle for the conducting region, together with the total field equality principle in the dielectric region, renders the problem into one which can be solved in free-space. As a result of this, the use of the particular Green's function and the associated approximations are avoided. The expansion functions are piecewise sinusoidal functions and unit pulses for conductor and dielectric regions respectively. Conductor basis functions are also used in the testing of the IE leading to a Galerkin type formulation for the conductor parts of the antenna. In the dielectric region, point matching is chosen to simplify the analysis.

The model is incorporated into a MoM code which can analyze LTSA's in air or

on a dielectric substrate, with a detector diode at the apex, or with a microstripline-to-slotline transition in the feed section. The code results are compared favorably to measurements and to available data in the literature. In order to check the approximation of the antenna taper by unequal size rectangular sectioning, the model is extended to analyze the skew-plate antenna and the results are favorably compared to a skew-segmentation model developed in this work. The variation of the radiation pattern with changing antenna parameters is investigated for the air TSA and the results are tabulated. It is observed that, narrower E-Plane beamwidths can be obtained as the antenna height, H, is reduced. Another important observation is that somewhat better antenna characteristics can be obtained for short antennas (small L), by reducing the antenna height (H) as well.

Since the matrix filling part of the MoM analysis is a major computational task, the computation time is reduced through the use of symmetry and the derivation of new simpler formulas for the mutual impedances of the perpendicular and parallel coplanar sinusoidal surface monopoles. Furthermore, the speed of the code is enhanced with vectorization of the matrix solution part of the algorithm, which employs a conjugate gradient iteration.

The model predicts the radiation characteristics of the LTSA with good accuracy. The unequal size rectangular sectioning scheme is a suitable approach converging to correct results by using approximately six segments per wavelength across the length and four to five segments per wavelength across the height of the antenna.

The computation times are realizable even for the largest antenna analyzed. The largest CPU time that is consumed by the code was about 2400 CPU seconds for $20\lambda_0^2$ air antenna, whereas the largest dielectric supported antenna ($15\lambda_0^2$) analysis consumed 3700 CPU seconds on CRAY Y-MP. Solvable problem size reduces with increasing dielectric permittivity, since more segments are required in the dielectric region because of the reduced slotline wavelength.

Suggestions for future work can be stated as:

- The CPU time requirements of the code can further be reduced by employing a table look-up algorithm in the matrix filling part of the code. This will also allow solution of larger structures.

- Another extension which can make possible the analysis of even larger problems is the employment of inhomogeneous sectioning for different parts of the antenna. The rectangular sectioning used in this work is homogeneous in the sense that the grid lines are equidistant. This approach is very simple and easy to implement as a computer code in terms of the identification of currents and the symmetry search for the impedance matrix calculations. However, not all parts of the antenna require the same grid for the same accuracy. For example, in the feed part of a microstripline fed antenna, more sections are required for both conductor and dielectric regions since the field is varying rapidly. Solution accuracy in this part of the antenna has a more important effect on the overall

solution compared to the sections farther away in the tapered region, since the wavelength in the slotline becomes larger. The same considerations also apply to the segments right near the taper and away from it. Therefore, the use of smaller sections in the feed part of the antenna and in the regions neighboring the taper, and larger sections elsewhere will yield smaller matrix sizes compared to homogeneous sectioning. However, the effect of this approach on the overall impedance matrix filling time should be studied.

- Only LTSA's are considered in this work. However, the developed model can be extended to handle other antenna structures as well. A natural extension of this work would be to analyze exponentially tapered slot antennas which have similar characteristics to the LTSA. A comparison between the two antennas with a parametric study (such as the one carried out in this work) would be very useful to the designers in the field. In particular, the cross-polarization level comparison can be very important, since the cross-polarization of the LTSA is quite high. Another modified structure of interest is a bi-slotline antenna which consists of two conducting sheets each having the same geometry as the single TSA, separated by a dielectric stub. This antenna can also be fed by a microstripline-to-bi-slotline transition. However, since the microstrip feed is contained in the structure, better sidelobe and and cross-polarization levels can be obtained.

- Finally, with further modifications, the code can be specialized to other antenna types such as printed bi-conical antennas (provided that the antenna edges make small angles), and microstrip antennas.

REFERENCES

[1] P.J.Gibson, "The Vivaldi Aerial," in *Proc. 9th European Microwave Conf.* (Brighton, U.K.), 1979, pp. 101-105.

[2] S.N.Prasad and S.Mahapatra, "A Novel MIC Slot-Line Antenna," in *Proc. 9th European Microwave Conf.* (Brighton, U.K.), 1979, pp. 120-124.

[3] K.S.Yngvesson, at al., "Endfire Tapered Slot Antennas on Dielectric Substrates," *IEEE Trans. Antennas Propagat.*, vol. AP-33, pp. 1392-1400, Dec. 1985.

[4] R.Janaswamy, "Radiation Pattern Analysis of the Tapered Slot Antenna," Ph.D. dissertation, Univ. Massachusetts, 1986.

[5] R.Janaswamy, Personal Communication, 1991.

[6] R.Janaswamy, D.H.Schaubert and D.M.Pozar, "Analysis of the Transverse Electromagnetic Mode Linearly Tapered Slot Antenna," *Radio Science*, vol. 21, pp. 797-804, Oct. 1986.

[7] R.Janaswamy and D.H.Schaubert, "Analysis of the Tapered Slot Antenna," *IEEE Trans. Antennas Propagat.*, vol. AP-35, pp. 1058-1064, Sept. 1987.

[8] K.S.Yngvesson, at al., "A New Integrated Slot Element Feed Array for Multibeam Systems," *IEEE Trans. Antennas Propagat.*, vol. AP-34, pp. 1372-1376, Nov. 1986.

[9] J.F.Johansson, "A Moment Method Analysis of Linearly Tapered Slot Antennas," *AP-S/URSI Conference Proc.*, pp. 383-387, June 1989.

[10] R.Janaswamy, "An Accurate Moment Method Model for the Tapered Slot Antenna," *IEEE Trans. Antennas Propagat.*, vol. AP-37, pp. 1523-1528, Dec. 1989.

[11] U.Kotthaus and B.Vowinkel, "Investigation of Planar Antennas for Submillimeter Receivers," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-37, pp. 375-380, Feb. 1989.

[12] K.S.Yngvesson at al, "The Tapered Slot Antenna - A New Integrated Element for Millimeter-Wave Applications," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-37, pp. 365-374, Feb. 1989.

[13] R.L.Carrel, "The Characteristic Impedance of Two Infinite Cones of Arbitrary Cross Section," *IRE Trans. Antennas Propagat.*, vol. AP-6, pp. 197-201, May 1958.

[14] S.A.Schelkunoff and H.T.Friis, *Antennas, Theory and Practice*,New York: Wiley, 1952.

[15] R.Janaswamy and D.H.Schaubert, "Characteristic Impedance of a Wide Slotline on Low-Permittivity Substrates," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-34, pp. 900-902, Aug. 1986.

[16] A.Köksal and J.F.Kauffman, "Moment Method Analysis of Linearly Tapered Slot Antennas," *IEEE AP-S URSI Symposium Digest*, vol. 1, pp. 314-317, June 24-28, 1991, London, Ontario-Canada

[17] E.K.Miller, "A Selective Survey of Computational Electromagnetics," *IEEE Trans. Antennas Propagat.*, vol. AP-36, pp. 1281-1305, Sept. 1988.

[18] B.H.McDonald and A.Wexler, "Finite-Element Solution of Unbounded Field Problems," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-20, pp. 841-847, Dec. 1972.

[19] R.F.Harrington, "Matrix Methods for Field Problems," *Proc. IEEE* , vol. 55, pp. 136-149, Feb. 1967.

[20] R.F.Harrington, *Field Computation by Moment Methods*, New York: Macmillan, 1968.

[21] T.K.Sarkar, "The Conjugate Gradient Technique as Applied to Electromagnetic Field Problems," *IEEE AP-S Newsletter*, vol. 28, No. 4, pp. 5-14, 1986.

[22] T.K.Sarkar, E.Arvas and S.M.Rao, "Applications of FFT and Conjugate Gradient Method for the Solution of Electromagnetic Radiation from Electrically Large and Small Conducting Bodies," *IEEE Trans. Antennas Propagat.*, vol. AP-33, pp. 635-640, 1986.

[23] M.F.Catedra, J.A.Alcaraz and J.C.Arredondo, "Analysis of Arrays of Vivaldi and LTSA Antennas," *AP-S/URSI Conference Proc.*, pp. 123-124, June 1989.

[24] D.M.Pozar, "Input Impedance and Mutual Coupling of Rectangular Microstrip Antennas," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 1191-1196, Nov. 1982.

[25] N.N.Wang, J.H.Richmond and M.C.Gilreath, "Sinusoidal Reaction Formulation for Radiation and Scattering from Conducting Surfaces," *IEEE Trans. Antennas Propagat.*, vol. AP-23, pp. 376-382, May 1975.

[26] D.E.Livesay and K.Chen, "Electromagnetic Fields Induced Inside Arbitrarily Shaped Biological Bodies," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp. 1273-1280, Dec 1974.

[27] T.K.Sarkar, E.Arvas and S.Ponnapalli, "Electromagnetic Scattering from Dielectric Bodies," *IEEE Trans. Antennas Propagat.*, vol. AP-37, pp. 673-676, May 1989.

[28] E.H.Newman and P.Tulyathan, "Wire Antennas in the Presence of a Dielectric/Ferrite Inhomogeneity," *IEEE Trans. Antennas Propagat.*, vol. AP-26, pp. 587-593, July 1978.

[29] T.K.Sarkar and E.Arvas, "An Integral Equation Approach to the Analysis of Finite Microstrip Antennas: Volume/Surface Formulation," *IEEE Trans. Antennas Propagat.*, vol. AP-38, pp. 305-312, March 1990.

[30] A.W.Glisson and D.R.Wilton, "Simple and Efficient Numerical Methods for Electromagnetic Radiation and Scattering from Surfaces," *IEEE Trans. Antennas Propagat.*, vol. AP-28, pp.593-603, Sept. 1980.

[31] S.M.Rao, D.R.Wilton and A.W.Glisson, "Electromagnetic Scattering by Surfaces of Arbitrary Shape," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp.409-418, May 1982.

[32] J.Singh and A.T.Adams, "A Nonrectangular Patch Model for Scattering from Surfaces," *IEEE Trans. Antennas Propagat.*, vol. AP-27, pp. 531-535, July 1979.

[33] E.H.Newman and P.Tulyathan, "A Surface Patch Model for Polygonal Plates," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 588-593, July 1982.

[34] E.H.Newman, P.Alexandropoulos and E.K.Walton, "Polygonal Plate Modeling of Realistic Structures," *IEEE Trans. Antennas Propagat.*, vol. AP-32, pp. 742-747, July 1984.

[35] D.L.Knepp and J.Goldhirsh, "Numerical Analysis of Electromagnetic Radiation Properties of Smooth Conducting Bodies of Arbitrary Shape," *IEEE Trans. Antennas Propagat.*, vol. AP-19, pp. 383-388, May 1972.

[36] E.H.Newman and D.M.Pozar, "Electromagnetic Modeling of Composite Wire and Surface Geometries," *IEEE Trans. Antennas Propagat.*, vol. AP-26, pp.784-789, Nov. 1978.

[37] H.E.King, "Mutual Impedances of Unequal Length Antennas in Echelon," *IRE Trans. Antennas Propagat.*, vol. AP-5, pp. 306-313, July 1957.

[38] J.H.Richmond and N.H.Geary, "Mutual Impedance Between Coplanar-Skew Dipoles," *IEEE Trans. Antennas Propagat.*, vol. AP-18, pp. 414-416, May 1970.

[39] E.H.Newman and D.M.Pozar, "Considerations for Efficient Wire/Surface Modeling," *IEEE Trans. Antennas Propagat.*, vol. AP-28, pp. 121-125, Jan. 1980.

[40] R.Janaswamy, "A Simplified Expression for the Self/Mutual Impedance between Coplanar and Parallel Surface Monopoles," *IEEE Trans. Antennas Propagat.*, vol. AP-35, pp. 1174-1176, Oct. 1987.

[41] G.Davis, "An Expression for the Mutual Impedance of Coplanar, Orthogonal Surface Patches," *IEEE Trans. Antennas Propagat.*, vol. AP-38, pp. 1975-1978, Dec. 1990.

[42] A.Köksal and J.F.Kauffman, "Mutual Impedance of Parallel and Perpendicular Coplanar Surface Monopoles,", *IEEE Trans. Antennas Propagat.*, vol. AP-39, pp. 1251-1256, Aug. 1991.

[43] D.H.Schaubert, D.R.Wilton and A.W.Glisson, "A Tetrahedral Modeling Method for Electromagnetic Scattering by Arbitrarily Shaped Inhomogeneous Dielectric Bodies," *IEEE Trans. Antennas Propagat.*, vol. AP-32, pp. 77-85, Jan. 1984.

[44] M.C.Bailey and M.D.Desphande, "Integral Equation Formulation of Microstrip Antennas," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 651-656, July 1982.

[45] M.D.Desphande and M.C.Bailey, "Input Impedance of Microstrip Antennas," *IEEE Trans. Antennas Propagat.*, vol. AP-30, pp. 645-651, July 1982.

[46] D.M.Pozar and S.M.Voda, "A Rigorous Analysis of a Microstripline Fed Patch Antenna," *IEEE Trans. Antennas Propagat.*, vol. AP-35, pp. 1343-1349, Dec. 1987.

[47] A.J.Poggio and E.K.Miller, "Integral equation solutions of three-dimensional scattering problem, in" *Techniques for Electromagnetics* , R.Mittra, Ed. Elmsford, NY: Pergamon, 1973

[48] A.W.Glisson, "An Integral Equation for Electromagnetic Scattering from Homogeneous Dielectric Bodies," *IEEE Trans. Antennas Propagat.*, vol. AP-32, pp. 173-175, Feb. 1984.

[49] A.D.Yaghjian, "Electric Dyadic Green's Functions in the Source Region," *Proceedings of the IEEE*, vol. 68, pp. 248-263, Feb. 1980.

[50] J.B.Knorr, "Slot-Line Transitions," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-22, pp. 548-554, May 1974.

[51] M.R.Hestenes and E.Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *J. Res. Nat. Bur. Standards*, vol. 49, pp. 409-436, Dec.1952.

[52] J.W.Daniel, "The Conjugate Gradient Method for Linear and Nonlinear Operator Equations," *SIAM J. Numer. Anal.*, vol. 4, No.1, pp. 10-26, 1967.

[53] T.K.Sarkar, "On the Application of the Generalized BiConjugate Gradient Method," *Journal of Electromagn. Waves and App.*, vol. 1, No. 3, pp. 223-242, 1987.

[54] T.K.Sarkar, K.R.Siarkiewicz and R.F.Stratton, "Survey of Numerical Methods for Solution of Large Systems of Linear Equations for Electromagnetic Field Problems," *IEEE Trans. Antennas Propagat.*, vol. AP-29, pp. 847-856, Nov. 1981.

[55] T.K.Sarkar and E.Arvas, "On a Class of Finite Step Iterative Methods (Conjugate Directions) for the Solution of an Operator Arising in Electromagnetics," *IEEE Trans. Antennas Propagat.*, vol. AP-33, pp. 1058-1066, Oct. 1985.

[56] A.C.Ludwig, "The Definition of Cross Polarization," *IEEE Trans. Antennas Propagat.*, vol. AP-21, No. 1, pp. 116-119, Jan. 1973.

Appendix A

USER GUIDE FOR LTSA:

MoM Analysis Code for

Linearly Tapered Slot Antennas

Appendix A

USER GUIDE FOR LTSA:

MoM Analysis Code for

Linearly Tapered Slot Antennas

A.1 Introduction

In this part, the preparation of the executable program of the MoM code written for the analysis of LTSA's is explained. The preparation of the input files with explanation of the input variables are described. The code listing is also included for easy reference in Section A.5.

The code described here is compatible with Unicos 6.0 Fortran 77, which is supported by CRAY Y-MP of the North Carolina Supercomputing Center. However, language extensions are avoided to make transition to other systems easy.

A.2 Preparation of ltsa

In order to prepare the executable, all of the routines included in the **makefile** (See Section A.5) should be placed in one directory. Entering the command

make

in the same directory prepares and maintains the executable program. This method is also suitable for further modifications and development of the code because of the easiness of the maintenance.

A.3 Running ltsa and pattern programs

After the executable program is prepared, the preparation of input file has to be carried out which is explained in the next section. With an input file **inp**, and a desired output file **out**, the program is run by directing the default input and output to the files as follows:

C-2

ltsa < inp > out

At the end of execution, the program writes the unprepared data (unformatted sequential output is used to save CPU time) to the file **out**. Before the pattern calculation programs **pattern** and **patdd** can be run, this data should be organized using the organization program **org**. This program uses the file **out** as its input and generates the files **cur** and **dat**. The file **cur** contains the current densities for the conductor and the dielectric regions, whereas **dat** contains the segmentation data and other input variables which have to be carried to the pattern programs for the completion of the analysis. The program **org** is run with the following command.

org < out

The pattern programs are written in interactive fashion, that is, the user is required to enter the names of the input and output files and the number of data points in pattern calculations.

A.4 Input File Organization and Variables

The organization of the input file is as follows,

```
figair, figdip, figms
w,      wf,     lo
li,     ls,     lssc
fiang,  ncw,    mi
lsc,    wsc
xd0,    lend,   hghd,  er
nld,    nwd,    nhd
wq,     lf,     nf
```

No special formatting is required for the data, it is entered in free format. All lengths are required in terms of the free space wavelength at the frequency of oper-

ation. The descriptions of the input variables are given below. Whenever a variable is not applicable for the required analysis zero should be entered in its place unless otherwise stated.

Input Line 1

flgair, flgdip and flgms determine the antenna type that will be analyzed. Different feeding options are also possible. The following are the currently available options.

- flgair=1, flgdip=1, flgms=0.

  This choice of variables results in an analysis of air LTSA, with a receiver diode soldered at the apex

- flgair=0, flgdip=1, flgms=0

  This set is used for dielectric LTSA's with a receiver diode at the apex.

- flgair=0, flgdip=0, flgms=0

  This set corresponds to dielectric LTSA's with an input transition part consisting of a microstripline to slotline transition. However, the microstripline is modeled by an infinitesimal current element for this choice of variables. This source modeling gives accurate results in the E and H plane of the antenna and for the D plane copolar radiation pattern. It cannot predict the correct cross polarization level though, since its choice results in a symmetric structure about x=0 (See Chapter 4).

- flgair=0, flgdip=0, flgms=1

  This set is used in the analysis of dielectric LTSA's with microstripline transition. In this case the microstripline current is also included among the unknowns.

Input Line 2

- w: w is the height of the antenna (H) in the other parts of the report.
- wf: Apex width of the antenna. In other words, wf is the slotline width where the antenna taper starts.
- lo: Length of the tapered part of the antenna.

Input Line 3

- li: Length of the feeding section of the antenna, not applicable when flgdip=1.
- ls: ls is the distance of source from the apex. For microstripline it is measured from the center of the microstripline. Not applicable when flgdip=1.
- lssc: lssc is the distance of source from the slotline short circuit. For microstripline it is measured from the slot short circuit edge to the microstripline edge. Not applicable when flgdip=1.

Input Line 4

- flang: Half taper angle of the LTSA in degrees.
- ncw: Number of segments across the height of the antenna for the conductor parts. Note that H measures the height of only one plate of the antenna, not the total height.
- mi: Number of segments across the length of the antenna when flgdip=1. When flgair=0, flgdip=0, it becomes the number of segments across the feeding section of the antenna. The number of segments in the tapered part for this case is calculated in the program using this variable.

Input Line 5

- lsc : half length of the receiving diode when flgdip=1. When flgair=0 and flgdip=0, it is the half length of the slotline short circuit current flowing between the lower and upper plates of the antenna.
- wsc: Half width of the receiving diode when flgdip=1, enter 0 otherwise.

Input Line 6

- xd0: Distance between the conductor edge and the dielectric edge in x direction if the two does not extend the same length. It is measured as $(x_{dielectric} - x_{conductor})$.
- lend: Length of the dielectric in x direction.
- hghd: Thickness of the dielectric region (y direction).
- er: Relative permittivity of the dielectric substrate.

Input Line 7

- nld: Number of segments in x direction for the dielectric substrate.
- nwd: Number of segments for the total width of the dielectric substrate.
- nhd: Number of segments across the thickness for the dielectric region. Currently the approximations in the code is written for nhd=1 case. Therefore, nhd=1 should be entered in the input file.

Input Line 8

- wq: Width of the microstripline, not applicable when flgms=0.
- lf: Length of the microstripline, not applicable when flgms=0.
- nf: Number of segments along the microstrip, not applicable when flgms=0. (nf=1 results after the execution when flgms=0, to account for the receiving diode.)

In the next section, the listings of the routines used in this work will be given. Explanation of the function of each routine is given at the beginning of the routines.

A.5 Listing of Programs

MAKEFILE

```
OBJS= mom.o struc.o fillcc.o filvlt.o decide.o par.o orthog.o \
      fillcd.o filldc.o filldd.o zxx.o zxy.o zxz.o zyy.o zyz.o \
      zzz.o zcdxx.o zcdxy.o zcdxz.o zcdzx.o zcdzy.o zcdzz.o \
      zdcxy.o zdcxz.o zdczx.o zdczy.o zdczz.o zdcxx.o \
      gaus24.o gaus6.o gaus4.o gaus2.o symxx.o symxy.o symxz.o \
      symyy.o symyz.o symzz.o partot.o ortot.o parf.o orf.o \
      parftot.o orftot.o filvltn.o
ltsa: $(OBJS) cgrad.o
cf77 $(OBJS) cgrad.o -o tsav
struc.o: struc.f
mom.o: mom.f
cgrad.o: cgrad.f
 cf77 -Zv -c -Wf"-em" cgrad.f
filvlt.o: filvlt.f
filvltn.o: filvltn.f
fillcc.o: fillcc.f
fillcd.o: fillcd.f
filldc.o: filldc.f
filldd.o: filldd.f
decide.o: decide.f
par.o: par.f
parf.o: parf.f
partot.o: partot.f
parftot.o: parftot.f
orthog.o: orthog.f
orf.o: orf.f
ortot.o: ortot.f
orftot.o: orftot.f
zxx.o: zxx.f
zxy.o: zxy.f
zxz.o: zxz.f
zyy.o: zyy.f
zyz.o: zyz.f
zzz.o: zzz.f
zcdxx.o: zcdxx.f
zcdxy.o: zcdxy.f
zcdxz.o: zcdxz.f
zcdzx.o: zcdzx.f
zcdzy.o: zcdzy.f
zcdzz.o: zcdzz.f
zdcxx.o: zdcxx.f
zdcxy.o: zdcxy.f
zdcxz.o: zdcxz.f
zdczx.o: zdczx.f
zdczy.o: zdczy.f
zdczz.o: zdczz.f
gaus24.o: gaus24.f
gaus6.o: gaus6.f
gaus4.o: gaus4.f
gaus2.o: gaus2.f
symxx.o: symxx.f
symxy.o: symxy.f
symxz.o: symxz.f
symyy.o: symyy.f
symyz.o: symyz.f
symzz.o: symzz.f
```

```
      program struc main
      real lngth,wdth,wf,pi,ma,expo,lcx,wu(2000),hu(2000),
     +        lcz,a,b,flang,ls,li,
     +        xs0,lo,wsc,lsc,lssc,er,wq,hf,lf,
     +        widd,hghd,ldx,ldy,ldz,lend,freq,xd0
      integer ii,jj,kj,ncl,ncw,cnt,j,m1,m2,mp,mq,nf,
     +        lin,dum2(2000),elnum,cntu,dum(2000),mi,nclo,n,
     +        nld,nhd,nwd,elnumd,md1,md2,flgair,flgdip,flgms
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Main program struc reads the input data, makes the                 c
c   segmentation, numbers the unknowns, calculates the sizes of        c
c   segments.                                                          c
c                                                                      c
c   calls: mom                                                         c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   read the input variables
c
      read(5,*) flgair,flgdip,flgms
      read(5,*) w,wf,lo
      read(5,*) li,ls,lssc
      read(5,*) lsc,wsc
      read(5,*) xd0,lend,hghd,er
      read(5,*) nld,nwd,nhd
      read(5,*) wq,lf,nf
      flgdip=1
      flgms=0
      pi=atan(1.0)*4.0
c
c     Antenna is represented by the addition of linear and exp.
c     terms; depending on antenna type one of the terms becomes zero
c     and the other is used. Only linear taper is analyzed so lin=1
c
      lin=1
      if (lin.eq.1) then
      expo=0.0
      a=0.0
      b=1.0
      ma=tan(flang*pi/180.0)
      else
      a=1.0
      ma=0.0
      b=0.0
      endif
      if (flgdip.eq.0) then
      lcx=li/(mi)
      nclo=int(lo/lcx)
      if ((lo-(nclo)*lcx).gt.(lcx/2.0)) nclo=nclo+1
      xs0=li-ls
      wsc=(xs0-lssc)/2.0
      ncl=mi+nclo
      lngth=(ncl)*lcx
      else
      ncl=mi
      lcx=lo/ncl
      lngth=lo
      li=0.0
      xs0=0.0
      mi=0
      endif
c
      flang=flang*pi/180.0
      lcz=wdth/(ncw)
      m1=ncl*ncw
      mp=2*ncw
      mq=2*ncw+ncl
      cnt=1
c
c   make the segmentation for the lower plate
```

```
c
      do 10 j=1,m1
      dum(j)=0
      wu(j)=lcx
      hu(j)=lcz
c
c   call decide to check if segment is in the geometry, if it is
c   calculate the segment sizes
c
      call decide(j,lngth,wdth,wf,ma,expo,a,b,lcx,lcz,ncl,ncw,
     +            dum(j),hu(j),cntu,li)
      wu(j)=lcx
c
      jj=int((j-1)/ncl)+1
      kj=j-(jj-1)*ncl
      if (flgdip.eq.1) then
      if (jj.eq.ncw) then
      if (kj.eq.(mi+1)) then
      dum(j)=1
      wu(j)=lcx
      hu(j)=lcz
      endif
      endif
      endif
  10  continue
c
c   make the segmentation for the lower plate
c
      do 15 j=m1+1,2*m1
      m2=int((j-ncl*ncw-1)/ncl)+1
      dum(j)=dum(j-(2*m2-1)*ncl)
  15  continue
c
c   repeat the segmentation again to count the vertical currents
c
      do 17 j=2*m1+1,4*m1
      dum(j)=dum(j-2*m1)
  17  continue
c
      cntu=cntu-1
c
c   number the x-directed currents for the lower antenna plate
c
      do 20 j=1,m1
      jj=int((j-1)/ncl)+1
      ii=j-(jj-1)*ncl
      if (ii.eq.ncl) then
      dum2(j)=0
      goto 20
      endif
      if ((dum(j).eq.0).or.(dum(j+1).eq.0)) then
      dum2(j)=0
      else
      dum2(j)=cnt
      cnt=cnt+1
      endif
  20  continue
c
c   number the x-directed currents for the upper antenna plate
c
      do 23 j=m1+1,2*m1
      dum2(j)=0
      m2=int((j-m1-1)/ncl)+1
      jj=j-(2*m2-1)*ncl
      if (dum2(jj).ne.(0)) then
      dum2(j)=cnt
      cnt=cnt+1
```

```
      endif
   23 continue
c
c    number the z-directed currents for the lower antenna plate
c
      do 25 j=2*m1+1,3*m1
      dum2(j)=0
      jj=int((j-2*m1-1)/ncl)+1
      if (jj.eq.ncw) goto 25
      if ((dum(j).eq.0).or.(dum(j+ncl).eq.0)) goto 25
      dum2(j)=cnt
      cnt=cnt+1
   25 continue
c
c    number the z-directed currents for the upper antenna plate
c
      do 27 j=3*m1+1,4*m1
      dum2(j)=0
      jj=int((j-3*m1-1)/ncl)+1
      if (jj.eq.ncw) goto 27
      if ((dum(j).eq.0).or.(dum(j+ncl).eq.0)) goto 27
      dum2(j)=cnt
      cnt=cnt+1
   27 continue
      cnt=cnt-1
c
c    cnt is the number of elements
c
      elnum=cnt
      if (flgms.eq.1) then
      hf=lf/nf
      else
      wq=0.0
      hf=0.0
      lf=0.0
      nf=1
      endif
      do 29 j=1,nf
      dum2(4*m1+j)=elnum+j
   29 continue
      elnum=elnum+nf
      n=elnum
c
      elnumd=0
      md1=0
      md2=0
      if (flgair.eq.1) goto 126
c
c    calculate the number of unknowns
c    for the dielectric part of the antenna
c
      widd=2.0*wdth+wf
      md1=nld*nwd
      md2=md1*nhd
      ldx=lend/(nld)
      ldy=hghd/(nhd)
      ldz=widd/(nwd)
      elnumd=3*md2
c
  126 continue
      n=elnum+elnumd
c
c    call the controlling routine -- mom
c
      call mom(lngth,wdth,flang,lcz,lcx,wf,hu,wu,xs0,wsc,lsc,ncl,
     +         ncw,m1,dum,dum2,elnum,n,er,mi,lend,widd,hghd,nld,
     +         nhd,nwd,ldx,ldy,ldz,elnumd,md1,md2,freq,xd0,
     +         flgair,flgdip,flgms,wq,hf,nf)
      end
```

```
      subroutine decide (j,lngth,wdth,wf,ma,expo,a,b,w,h,n,ncw,
     +                   dum,hu,cntu,li)
      real  lngth,wdth,wf,ma,expo,a,b,w,h,zant1,xc1,xc2,
     +        zant2,zc1,zc2,hu,li
      integer j,jj,ii,n,ncw,dum,cntu
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine decide if the segment is inside the geometry, if it c
c   is, it calculates the height and length of it.                 c
c   z-directed current source                                      c
c                                                                  c
c   called by: struc                                               c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      jj=int((j-1)/n)+1
      ii=j-(jj-1)*n
      xc1=(ii)*w
      zc1=-((ncw-jj)*h+wf/2.0)
      zc2=zc1-h
      if (xc1.le.li) then
      zant1=-wf/2.0
      else
      zant1=-(a*((wf/2.0)*exp(expo*xc1))+
     +        b*(wf/2.0+ma*(xc1-li)))
      endif
      if (zant1.ge.zc1) dum=1
      xc2=(ii-1)*w
      if (xc2.le.li) then
      zant2=-wf/2.0
      else
      zant2=-(a*((wf/2.0)*exp(expo*xc1))+
     +        b*(wf/2.0+ma*(xc2-li)))
      endif
      if (zant2.lt.zc2) dum=0
      if ((zant2.gt.zc2).and.(zant2.le.zc1)) then
      if (zant1.ge.zc2) then
      hu=((zant2-zc2)+(zant1-zc2))/2.0
      dum=2
      else
      hu=((zant2-zc2)+(zant1-zc2))/2.0
      if (hu.lt.(-h)) then
      write(6,*) ' Increase ncl or decrease ncw'
      goto 775
      endif
      if (hu.le.(0.0)) then
      dum=0
      else
      dum=2
      endif
      endif
      endif
      if ((zant2.ge.zc1).and.(zant1.lt.zc1)) then
      if (zant1.lt.zc2) then
      write(6,*) ' increase ncl or decrease ncw'
      goto 775
      endif
      hu=((zant2-zc1)+(zant1-zc1))/2.0
      if (hu.ge.(0.0)) then
      dum=1
      hu=h
      else
      dum=2
      hu=h+hu
      endif
      endif
  775 continue
      if ((hu.eq.h).and.(dum.eq.2))  dum=1
      return
      end
```

```
      subroutine mom(lngth,wdth,flang,lcz,lcx,wf,hu,wu,xs0,wsc,lsc,
     +               ncl,ncw,k,dum,dum2,elnumc,n,er,mi,
     +               lend,widd,hghd,nld,nhd,nwd,lx,ly,lz,elnumd,
     +               md1,md2,freq,xd0,flgair,flgdip,flgms,wq,hf,nf)
      real xs0,lngth,wdth,flang,lcz,lcx,wf,wu(2000),xd0,
     +        wsc,lsc,hu(2000),er,wq,hf,
     +        lend,widd,hghd,lx,ly,lz,freq
      complex vv(n),imp(n,n),cr(n)
      integer i,ncw,ncl,k,dum2(2000),dum(2000),elnumc,n,mi,
     +        nld,nhd,nwd,elnumd,md1,md2,nu,flgair,flgdip,
     +        flgms,nf
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine  mom calls the necessary routines in the program     c
c   and outputs the analysis results                                c
c                                                                   c
c   called by: struc                                                c
c                                                                   c
c   calls: filvlt, filvltn, filcc, filcd, fildc, fildd, cgrad       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      if (flgdip.eq.1) then
      do 11 j=1,n
      vv(j)=0.0
  11  continue
      vv(elnumc)=1.0
      else
      if (flgms.eq.0) then
c
c   if current element at apex call filvlt
c
      call filvlt(dum,dum2,lngth,wdth,hghd,k,ncw,ncl,elnumc,n,lcz,
     +            lcx,wf,vv,wu,hu,xs0,wsc,lsc,elnumd,md1,md2,lx,
     +            ly,lz,xd0,nld)
      else
c
c   if microstripline feed call filvltn
c
      call filvltn(dum,dum2,lngth,wdth,hghd,k,ncw,ncl,elnumc,n,lcz,
     +            lcx,wf,vv,wu,hu,xs0,wsc,lsc,elnumd,md1,md2,lx,
     +            ly,lz,xd0,nld,wq,hf,nf)
      endif
      endif
c
c   fill the conductor-conductor interactions of the imp. matrix
c
      call filcc(dum,dum2,lngth,wdth,k,ncw,ncl,elnumc,lcz,
     +            lcx,wf,imp,wu,hu,xs0,wsc,lsc,n,hghd,er,mi,
     +            wq,hf,nf)
      if (flgair.eq.1) goto 76
c
c   fill the conductor-dielectric interactions of the imp. matrix
c
      call filcd(lcx,lcz,lx,ly,lz,imp,dum,dum2,k,ncl,ncw,md1,
     +            md2,nld,nwd,hghd,wdth,wf,elnumc,elnumd,
     +            wu,hu,xs0,wsc,lsc,n,mi,xd0,wq,hf,nf)
c
c   fill the dielectric-conductor interactions of the imp. matrix
c
      call fildc(lcx,lcz,lx,ly,lz,imp,dum,dum2,k,ncl,ncw,md1,
     +            md2,nld,nwd,hghd,wdth,wf,elnumc,elnumd,
     +            wu,hu,xs0,wsc,lsc,n,mi,xd0,wq,hf,nf)
c
c   fill the dielectric-dielectric interactions of the imp. matrix
c
      call fildd(lx,ly,lz,imp,nld,nhd,nwd,
     +            md1,md2,er,freq,elnumc,n)
  76  continue
      nu=n
```

```
      do 82 i=1,nu
      cr(i)=0.0
 82   continue
c
c    solve the matrix equation using CG method
c
      call cgrad(imp,vv,n,nu,cr)
c
c    output the analysis results
c
 111  continue
      write(6,*) flgair,flgdip,flgms
      write(6,*) ncl,ncw,lngth,wdth,wf,k,elnumc
      write(6,*) xs0,wsc,lsc,xd0,mi,flang
      write(6,*) elnumd,md1,md2,nld,nhd,nwd,hghd
      write(6,*) lx,ly,lz,freq,er
      write(6,*) wq,hf,nf
      write(6,*) dum2,dum,wu,hu
      write(6,*) cr
 344  continue
      return
      end
```

```
      subroutine filvlt(dum,dum2,lngth,wdth,t,mc1,ncw,ncl,elnumc,
     +                  n,lcz,lcx,wf,vv,wu,hu,xs0,wsc,lsc,elnumcd,md1,
     +                  md2,lx,ly,lz,xd0,nld)
      real lngth,wdth,wf,kwf,lcx,lcz,pi,kxs0,w1,h1,
     +        wu(2000),hu(2000),i1,i2,xs0,
     +        wsc,lsc,t,kt,lx,ly,lz,xd0,d,hh,l
      complex vv(n),z12,z,Ex,Ey,Ez
      integer mc1,ncw,ncl,xl,xu,yl,yu,ii,jj,i,dum2(2000),
     +        elnumc,dum(2000),jp,ip,n,md1,md2,elnumcd,it,ih,
     +        iw,il,nld,j,jxl,jxu,jyl,jyu
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filvlt computes voltage vector for infinitesimal    c
c   z-directed current source                                      c
c                                                                  c
c   called by: mom                                                 c
c                                                                  c
c   calls: Efld, zcdxz, zcdzz                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      pi=atan(1.0)*4.0
      kwf=2.0*pi*wf
      kt=2.0*pi*t
c
      do 10 j=1,4*mc1+1
      if (dum2(j).eq.0) goto 10
      vv(dum2(j))=0.0
      jxl=0
      jxu=0
      jyl=0
      jyu=0
c   determine the location of the patch
      if ((j.ge.0).and.(j.le.mc1)) jxl=1
      if ((j.gt.mc1).and.(j.le.(2*mc1))) jxu=1
      if ((j.gt.(2*mc1)).and.(j.le.(3*mc1))) jyl=1
      if ((j.gt.(3*mc1)).and.(j.le.(4*mc1))) jyu=1
      z12=0.0
c
      if (jxl.eq.1) then
c   if x-current in the lower antenna plate calculate
c   the z field of x directed conductor current to find the
c   voltage element
      jj=int((j-1)/ncl)+1
      h1=hu(j+1)*2.0*pi
      w1=lcx*pi
      d=(xs0-(j-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=kt
      l=(wdth+wf/2.0-(jj-1)*lcz)*2.0*pi
      i1=0.0
      i2=1.0
      call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kwf
      endif
c
      if (jxu.eq.1) then
c   use symmetry
      jj=int((j-mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl
      vv(dum2(j))=-vv(dum2(jp))
      endif
c
```

```
      if (j.eq.(4*mc1+1)) then
c   for the short-circuit calculate z field of z current
      w1=wsc*2.0*pi
      h1=lsc*2.0*pi
      d=(xs0-wsc)*2.0*pi
      hh=kt
      l=lsc*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kwf
      endif
c
      if (jyl.eq.1) then
c
c   if z-current in the lower antenna plate calculate
c   the z field of z directed conductor current to find the
c   voltage element
c
      jj=int((j-2*mc1-1)/ncl)+1
      jp=j-2*mc1
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(xs0-(jp-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=kt
      l=(wdth+wf/2.0-(jj-1)*lcz)*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp+ncl)*2.0*pi
      call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kwf
      endif
c
      if (jyu.eq.1) then
c   use symmetry for upper z-currents
      jj=int((j-3*mc1-1)/ncl)+1
      jp=j-2*jj*ncl
      vv(dum2(j))=vv(dum2(jp))
      endif
c
   10 continue
c
c   continue with the dielectric region
      do 27 i=1,md2
      it=int((i-1)/md2)
      ih=int((i-it*md2-1)/md1)
      iw=int((i-it*md2-ih*md1-1)/nld)
      il=i-it*md2-ih*md1-iw*nld
c   calculate the location of the diel. current
      d=(xd0+(il)*lx-lx/2.0-xs0)*2.0*pi
      hh=-((ih)*ly+ly/2.0)*2.0*pi
      l=((iw)*lz+lz/2.0-(wdth+wf/2.0))*2.0*pi
c   calculate the field of the source at the center
      call Efld(Ex,Ey,Ez,d,l,hh,kwf)
      vv(i+elnumc)=-Ex
      vv(i+elnumc+md2)=-Ey
      vv(i+elnumc+2*md2)=-Ez
```

```
  27   continue
       return
       end
c
       subroutine Efld(Ex,Ey,Ez,y,z,h,wf)
       real c1,c2,wf,y,z,kt,c5,h
       complex p1,Ex,Ey,Ez,c3,c4
c  calculates the field of an infinitesimal current element
c  at the point x=y, y=h, z=z
       p1=(0.0,1.0)
       c1=60.0*wf
       c2=sqrt(y**2+z**2+h**2)
       c3=c1/(c2**4)*(1.0-p1/c2)
       c4=p1*c1/(2.0*(c2**3))*(1.0-(p1/c2)-1.0/(c2**2))
       Ex=(c3+c4)*y*z*cexp(-p1*c2)
       Ey=(c3+c4)*h*z*cexp(-p1*c2)
       Ez=(c3*z*z-c4*(y*y+h*h))*cexp(-p1*c2)
c
       return
       end
```

```
      subroutine filvltn(dum,dum2,lngth,wdth,t,mc1,ncw,ncl,elnumc,
     +                   n,lcz,lcx,wf,vv,wu,hu,xs0,wsc,lsc,elnumcd,md1,
     +                   md2,lx,ly,lz,xd0,nld,wq,hf,nf)
      real lngth,wdth,wf,kwf,lcx,lcz,pi,kxs0,wi,hi,
     +        wu(2000),hu(2000),i1,i2,xs0,wq,hf,hfj,
     +        wsc,lsc,t,kt,lx,ly,lz,xd0,d,hh,l
      complex vv(n),z12,z,Ex,Ey,Ez
      integer mc1,ncw,ncl,xl,xu,yl,yu,ii,jj,i,dum2(2000),
     +        elnumc,dum(2000),jp,ip,n,md1,md2,elnumcd,it,ih,
     +        iw,il,nld,j,jxl,jxu,jyl,jyu,nf,jfl,kj
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filvltn computes voltage vector for infinitesimal c
c   y-current source located at  feed point of microstripline    c
c                                                                c
c   called by: mom                                               c
c                                                                c
c   calls: Efldn, zcdxy, zcdzy                                   c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      pi=atan(1.0)*4.0
      kwf=2.0*pi*wf
      kt=2.0*pi*t
c
      do 10 j=1,4*mc1+nf
      if (dum2(j).eq.0) goto 10
      vv(dum2(j))=0.0
      jxl=0
      jxu=0
      jyl=0
      jyu=0
      jfl=0
c   determine the location of the patch
      if ((j.ge.0).and.(j.le.mc1)) jxl=1
      if ((j.gt.mc1).and.(j.le.(2*mc1))) jxu=1
      if ((j.gt.(2*mc1)).and.(j.le.(3*mc1))) jyl=1
      if ((j.gt.(3*mc1)).and.(j.le.(4*mc1))) jyu=1
      if ((j.gt.(4*mc1+1)).and.(j.le.(4*mc1+nf))) jfl=1
      z12=0.0
c
      if (jxl.eq.1) then
c   if x-current in the lower antenna plate calculate
c   the y field of x directed conductor current to find the
c   voltage element
      jj=int((j-1)/ncl)+1
      h1=hu(j+1)*2.0*pi
      w1=lcx*pi
      d=(xs0-(j-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=kt/2.0
      l=-(jj-1)*lcz*2.0*pi
      i1=0.0
      i2=1.0
      call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
      if (jxu.eq.1) then
c   if x-current in the upper antenna plate calculate
c   the y field of x directed conductor current to find the
```

```
c   voltage element, no symmetry in this case
      jj=int((j-mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl
      kj=j-mc1-(jj-1)*ncl
      h1=hu(jp+1)*2.0*pi
      w1=lcx*pi
      l=-(jj*lcz+wdth+wf-hu(jp+1))*2.0*pi
      d=(xs0-((kj-1)+0.50)*lcx)*2.0*pi
      hh=kt/2.0
      i1=0.0
      i2=1.0
      call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
      if (j.eq.(4*mc1+1)) then
c   for the short-circuit calculate y field of z current
      w1=wsc*2.0*pi
      h1=lsc*2.0*pi
      d=(xs0-wsc)*2.0*pi
      hh=kt/2.0
      l=-(wdth+wf/2.0-lsc)*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
      if (jyl.eq.1) then
c
c   if z-current in the lower antenna plate calculate
c   the y field of z directed conductor current to find the
c   voltage element
c
      jj=int((j-2*mc1-1)/ncl)+1
      jp=j-2*mc1
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(xs0-(jp-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=kt/2.0
      l=-(jj-1)*lcz*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp+ncl)*2.0*pi
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
      if (jyu.eq.1) then
```

```
c
c   if z-current in the lower antenna plate calculate
c   the y field of z directed conductor current to find the
c   voltage element, no symmetry in this case
c
      jj=int((j-3*mc1-1)/ncl)+1
      kj=j-3*mc1-(jj-1)*ncl
      jp=j-(2*jj-1)*ncl-2*mc1
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(xs0-((kj-1)+0.50)*lcx)*2.0*pi
      hh=kt/2.0
      l=-(jj*lcz+wdth+wf-hu(jp))*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp-ncl)*2.0*pi
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
      if (jf1.eq.1) then
c
c   if z-current on the microstripline  calculate
c   the y field of z directed conductor current to find the
c   voltage element
c
      jj=j-4*mc1-1
      hfj=(jj-1)*hf
      w1=wq*pi
      h1=hf*2.0*pi
      d=0.0
      hh=-kt/2.0
      l=-hfj*2.0*pi
      i1=0.0
      i2=1.0
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      vv(dum2(j))=z12*kt
      endif
c
   10 continue
c
c   continue with the dielectric region
      do 27 i=1,md2
      it=int((i-1)/md2)
      ih=int((i-it*md2-1)/md1)
      iw=int((i-it*md2-ih*md1-1)/nld)
      il=i-it*md2-ih*md1-iw*nld
c   calculate the location of the diel. current
      d=(xd0+(il)*lx-lx/2.0-xs0)*2.0*pi
      hh=0.0
      l=(iw*lz+lz/2.0)*2.0*pi
c   calculate the field of the source at the center
      call Efldn(Ex,Ey,Ez,d,l,hh,kt)
      vv(i+elnumc)=-Ex
      vv(i+elnumc+md2)=-Ey
      vv(i+elnumc+2*md2)=-Ez
```

```
  27  continue
      return
      end
c
      subroutine Efldn(Ex,Ey,Ez,x,z,y,kt)
      real c1,kt,x,y,z,cx,cy,cz,r
      complex p1,Ex,Ey,Ez
c  calculates the field of an infinitesimal y-current element
c  at the point x=x, y=y, z=z
      p1=(0.0,1.0)
      c1=30.0*kt
      r=sqrt(x**2+y**2+z**2)
      cx=x/r
      cy=y/r
      cz=z/r
      Ex=-p1*c1*cexp(-p1*r)*cx*cy*(-1.0+3.0*p1/r*(1.0-p1/r))/r
      Ey=(1.0-cy**2+p1/r*(1.0-p1/r)*(3.0*cy**2-1))/r
      Ey=-p1*c1*cexp(-p1*r)*Ey
      Ez=-p1*c1*cexp(-p1*r)*cz*cy*(-1.0+3.0*p1/r*(1.0-p1/r))/r
c
      return
      end
```

```
      subroutine filcc(dum,dum2,lngth,wdth,m1,ncw,ncl,elnum,
     +                 lcz,lcx,wf,z,wu,hu,xs0,wsc,lsc,n,t,
     +                 er,mi,wq,hf,nf)
      real    lngth,wdth,wf,lcx,wu(2000),hu(2000),wsc,lsc,
     +        lcz,wi,h1,w2,h2,d1,hh1,d,hh,i11,i21,i12,i22,xs0,
     +        t,tl,er,pi,kt,kw1,kh1,kw2,kh2,kwf,hf,wq,
     +        hfj,hfi,h11,h21,h12,h22,w12,w22,
     +        xs1,xs2,l,kb
      complex z(n,n),z12,zs,p1
      integer n,ii,jj,ki,kj,ncl,ncw,j,i,jxl,jxu,jyl,jyu,fff,mi,
     +        ixl,ixu,iyl,iyu,m1,aj,ai,p,q,jfl,ifl,nf,
     +        dum(2000),dum2(2000),elnum,jp,ip,ma,
     +        mb,m,flg
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filcc computes the conductor-conductor                  c
c   interactions of the impedance matrix (submatrix A)                 c
c                                                                      c
c   called by: mom                                                     c
c                                                                      c
c   calls: ortot, partot, orftot, parftot                              c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      do 30 j=1,4*m1+nf
      do 30 i=1,4*m1+nf
      if ((dum2(j).eq.(0)).or.(dum2(i).eq.(0))) goto 30
      flg=1
c   initialize
      jxl=0
      jxu=0
      jyl=0
      jyu=0
      ixl=0
      ixu=0
      iyl=0
      iyu=0
      jfl=0
      ifl=0
      z12=0.0
      z12c=0.0
c   determine the location of the conductor currents
      if ((j.ge.0).and.(j.le.m1)) jxl=1
      if ((j.gt.m1).and.(j.le.(2*m1))) jxu=1
      if ((j.gt.(2*m1)).and.(j.le.(3*m1))) jyl=1
      if ((j.gt.(3*m1)).and.(j.le.(4*m1))) jyu=1
      if ((i.ge.0).and.(i.le.m1)) ixl=1
      if ((i.gt.m1).and.(i.le.(2*m1))) ixu=1
      if ((i.gt.(2*m1)).and.(i.le.(3*m1))) iyl=1
      if ((i.gt.(3*m1)).and.(i.le.(4*m1))) iyu=1
      if ((j.gt.(4*m1+1)).and.(j.le.(4*m1+nf))) jfl=1
      if ((i.gt.(4*m1+1)).and.(i.le.(4*m1+nf))) ifl=1
c
      if ((jxl.eq.1).and.(i.eq.(4*m1+1))) then
c   calculate interaction between lower plate x-directed
c   conductor current and short-circuit or receiving diode
      jj=int((j-1)/ncl)+1
      kj=j-(jj-1)*ncl
      w1=wsc
      h11=lsc
      h21=h11
      w12=lcx/2.0
      w22=w12
      h2=hu(j+1)
      hh=-(wdth-(jj-1)*lcz+wf/2.0-lsc)
      d=((kj-1)+0.50)*lcx-wsc
```

```
      call ortot(d,hh,w1,h11,h21,w12,w22,h2,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jxu.eq.1).and.(i.eq.(4*m1+1))) then
c   use symmetry for upper x-directed conductor currents and
c   short-circuit or receiving diode interaction
      jj=int((j-m1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl
      z(dum2(j),dum2(i))=-z(dum2(jp),dum2(i))
      endif
c
      if ((jxl.eq.1).and.(ifl.eq.1)) then
c   calculate interaction between lower plate x-directed
c   conductor current and microstripline current
      jj=int((j-1)/ncl)+1
      kj=j-(jj-1)*ncl
      ii=i-4*m1-1
      hfi=(ii-1)*hf
      w1=wq/2.0
      h11=hf
      h21=hf
      w12=lcx/2.0
      w22=w12
      h2=hu(j+1)
      hh=(jj-1)*lcz-hfi
      d=((kj-1)+0.50)*lcx-xs0
      call orftot(d,hh,-t,w1,h11,h21,w12,w22,h2,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jxu.eq.1).and.(ifl.eq.1)) then
c   calculate interaction between upper plate x-directed
c   conductor current and microstripline current
      jj=int((j-m1-1)/ncl)+1
      kj=j-m1-(jj-1)*ncl
      jp=j-(2*jj-1)*ncl
      ii=i-4*m1-1
      hfi=(ii-1)*hf
      w1=wq/2.0
      h11=hf
      h21=hf
      w12=lcx/2.0
      w22=w12
      h2=hu(jp+1)
      hh=jj*lcz+wdth+wf-hu(jp+1)-hfi
      d=((kj-1)+0.50)*lcx-xs0
      call orftot(d,hh,-t,w1,h11,h21,w12,w22,h2,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jyl.eq.1).and.(i.eq.(4*m1+1))) then
c   calculate interaction between lower plate z-directed
c   conductor current and short-circuit or receiving diode
      jj=int((j-2*m1-1)/ncl)+1
      kj=j-2*m1-(jj-1)*ncl
      jp=j-2*m1
      w1=wsc
      h11=lsc
      h21=lsc
      w2=wu(jp)/2.0
      h12=hu(jp)
      h22=hu(jp+ncl)
      hh=-(wdth-(jj-1)*lcz+wf/2.0-lsc)
```

```
      d=((kj-1)+0.50)*lcx-wsc
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jyu.eq.1).and.(i.eq.(4*m1+1))) then
c   use symmetry for upper z-directed conductor currents and
c   short-circuit or receiving diode interaction
      jj=int((j-3*m1-1)/ncl)+1
      jp=j-2*jj*ncl
      z(dum2(j),dum2(i))=z(dum2(jp),dum2(i))
      endif
c
      if ((j.eq.(4*m1+1)).and.(ifl.eq.1)) then
c   calculate interaction between microstripline current
c   and short-circuit or receiving diode
      ii=i-4*m1-1
      hfi=(ii-1)*hf
      w2=wq/2.0
      h12=hf
      h22=hf
      w1=wsc
      h11=lsc
      h21=lsc
      hh=-(wdth-hfi+wf/2.0-lsc)
      d=xs0-wsc
      call parftot(d,hh,t,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jyl.eq.1).and.(ifl.eq.1)) then
c   calculate interaction between lower plate z-directed
c   conductor current and microstripline current
      jj=int((j-2*m1-1)/ncl)+1
      kj=j-2*m1-(jj-1)*ncl
      jp=j-2*m1
      ii=i-4*m1-1
      hfi=(ii-1)*hf
      w1=wq/2.0
      h11=hf
      h21=hf
      w2=wu(jp)/2.0
      h12=hu(jp)
      h22=hu(jp+ncl)
      hh=(jj-1)*lcz-hfi
      d=((kj-1)+0.50)*lcx-xs0
      call parftot(d,hh,-t,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      if ((jyu.eq.1).and.(ifl.eq.1)) then
c   calculate interaction between upper plate z-directed
c   conductor current and microstripline current
      jj=int((j-3*m1-1)/ncl)+1
      kj=j-3*m1-(jj-1)*ncl
      jp=j-(2*jj-1)*ncl-2*m1
      ii=i-4*m1-1
      hfi=(ii-1)*hf
      w1=wq/2.0
      h11=hf
      h21=hf
      w2=wu(jp)/2.0
      h12=hu(jp)
      h22=hu(jp-ncl)
      hh=jj*lcz+wdth+wf-hu(jp)-hfi
```

```
      d=((kj-1)+0.50)*lcx-xs0
      call parftot(d,hh,-t,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c     if ((jfl.eq.1).and.(ifl.eq.1)) then
c   calculate interaction between microstripline currents
      if (j.le.i) then
      jj=j-4*m1-1
      ii=i-4*m1-1
      if ((jj.eq.ii).and.(jj.gt.1)) then
      z(dum2(j),dum2(i))=z(dum2(j-1),dum2(i-1))
      else
      hfj=(jj-1)*hf
      hfi=(ii-1)*hf
      hh=hfi-hfj
      d=0.0
      w1=wq/2.0
      w2=w1
      h11=hf
      h21=hf
      h12=hf
      h22=hf
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
      endif
      endif
c     if ((jfl.eq.1).and.(i.lt.j)) then
c   use symmetry if i<j
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c     if ((j.eq.(4*m1+1)).and.(i.lt.j)) then
c   use symmetry if i<j
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c     if ((j.eq.(4*m1+1)).and.(i.eq.j)) then
c   calculate self interaction of short-circuit
c   or receiving diode
      w1=wsc
      h11=lsc
      h21=lsc
      w2=wsc
      h12=lsc
      h22=lsc
      d=0.0
      hh=0.0
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c     if ((jxl.eq.1).and.(ixl.eq.1)) then
c   investigate interaction between lower plate x-directed
c   antenna conductor currents
      jj=int((j-1)/ncl)+1
      ii=int((i-1)/ncl)+1
      ki=i-(ii-1)*ncl
c   check various possibilities of symmetry
      if (j.gt.i) then
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      else
      if ((dum(j).eq.2).or.(dum(j+1).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+1).eq.2)) flg=0
c     if ((jj.gt.1).and.(ii.gt.1)) then
```

```
      aj=j-ncl
      ai=i-ncl
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+1).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
c
      if ((jj.eq.1).and.(j.gt.1)) then
      if (ki.lt.j) then
      aj=ki
      ai=i+j-ki
      else
      aj=j-1
      ai=i-1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+1).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
      endif
c
      if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower plate x-directed
c   antenna conductor currents
      w1=hu(j+1)/2.0
      h11=wu(j)
      h21=wu(j+1)
      w2=hu(i+1)/2.0
      h12=wu(i)
      h22=wu(i+1)
      d=(jj-ii)*lcz+w1-w2
      hh=-(j-ncl*(jj-ii)-i)*lcx-h12+h11
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      endif
cc
cc
      if ((jxu.eq.1).and.(ixu.eq.1)) then
c   use symmetry for interaction between upper plate x-directed
c   antenna conductor currents
      jj=int((j-m1-1)/ncl)+1
      aj=j-(2*jj-1)*ncl
      ii=int((i-m1-1)/ncl)+1
      ai=i-(2*ii-1)*ncl
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      endif
cc
cc
      if ((jxl.eq.1).and.(ixu.eq.1)) then
c   investigate interaction between lower and upper plate
c   x-directed antenna conductor currents
      jj=int((j-1)/ncl)+1
      ii=int((i-m1-1)/ncl)+1
      ki=i-m1-(ii-1)*ncl
      ip=i-(2*ii-1)*ncl
      if ((dum(j).eq.2).or.(dum(j+1).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+1).eq.2)) flg=0
```

```
c   check various possibilities of symmetry
      if ((jj.gt.1).and.(ii.gt.1)) then
      aj=j-ncl
      ai=i-ncl
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+1).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
c
      if ((jj.eq.1).and.(j.gt.1)) then
      if (ki.lt.j) then
      aj=ki
      ai=i+j-ki
      else
      aj=j-1
      ai=i-1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+1).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
c
      if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower and upper plate
c   x-directed antenna conductor currents
      w1=hu(j+1)/2.0
      h11=wu(j)
      h21=wu(j+1)
      w2=hu(ip+1)/2.0
      h12=wu(ip)
      h22=wu(ip+1)
      d=(jj-ii-ncw-1)*lcz-wf+w2+w1
      hh=-(j-ncl*(jj-ii)-i+m1)*lcx-h12+h11
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      endif
c
c
c
      if ((jxu.eq.1).and.(ixl.eq.1)) then
c   use symmetry for interaction between upper plate x-directed
c   antenna conductor currents with those of lower plate
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
c
c
      if ((jyl.eq.1).and.(iyl.eq.1)) then
c   investigate interaction between lower plate z-directed
c   antenna conductor currents
      ii=int((i-2*m1-1)/ncl)+1
      jj=int((j-2*m1-1)/ncl)+1
      ki=i-2*m1-(ii-1)*ncl
      kj=j-2*m1-(jj-1)*ncl
      jp=j-2*m1
      ip=i-2*m1
c   check various possibilities of symmetry
      if (j.gt.i) then
```

```
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      else
c
      if ((dum(j).eq.2).or.(dum(j+ncl).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+ncl).eq.2)) flg=0
c
      if ((jj.gt.1).and.(ii.gt.1)) then
      aj=j-ncl
      ai=i-ncl
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+ncl).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
c
      if ((jj.eq.1).and.(kj.gt.1)) then
      if (ki.lt.kj) then
      aj=ki+2*m1
      ai=i+kj-ki
      else
      aj=j-1
      ai=i-1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+ncl).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
      endif
c
      if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower plate z-directed
c   antenna conductor currents
      w1=wu(jp)/2.0
      h11=hu(jp)
      h21=hu(jp+ncl)
      w2=wu(ip)/2.0
      h12=hu(ip)
      h22=hu(ip+ncl)
      d=(ki-kj)*lcx-w2+w1
      hh=(ii-jj)*lcz
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      endif
c
cc
c
      if ((jyl.eq.1).and.(iyu.eq.1)) then
c   investigate interaction between lower and upper plate
c   z-directed antenna conductor currents
      jj=int((j-2*m1-1)/ncl)+1
      ii=int((i-3*m1-1)/ncl)+1
      ki=i-3*m1-(ii-1)*ncl
      kj=j-2*m1-(jj-1)*ncl
      jp=j-2*m1
      ip=i-(2*ii-1)*ncl-2*m1
      if ((dum(j).eq.2).or.(dum(j+ncl).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+ncl).eq.2)) flg=0
      if ((jj.gt.1).and.(ii.gt.1)) then
c   check various possibilities of symmetry
```

```
      aj=j-ncl
      ai=i-ncl
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+ncl).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
c
      if ((jj.eq.1).and.(kj.gt.1)) then
      if (ki.lt.kj) then
      aj=ki+2*m1
      ai=i+kj-ki
      else
      aj=j-1
      ai=i-1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+ncl).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
c
      if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower and upper plate
c   z-directed antenna conductor currents
      w1=wu(jp)/2.0
      h11=hu(jp)
      h21=hu(jp+ncl)
      w2=wu(ip)/2.0
      h12=hu(ip)
      h22=hu(ip-ncl)
      d=(ki-kj)*lcx-w2+w1
      hh=(ii+ncw+1-jj)*lcz+wf-h12
      call partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      endif
c
cc
c
      if ((jyu.eq.1).and.(iyl.eq.1)) then
c   use symmetry for interaction between upper plate
c   and lower plate z-directed antenna conductor currents
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
ccc
c
      if ((jxl.eq.1).and.(iyl.eq.1)) then
c   investigate interaction between lower plate x-directed
c   and lower plate z-directed antenna conductor currents
      jj=int((j-1)/ncl)+1
      ii=int((i-2*m1-1)/ncl)+1
      ki=i-2*m1-(ii-1)*ncl
      kj=j-(jj-1)*ncl
      ip=i-2*m1
      if ((dum(j).eq.2).or.(dum(j+1).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+ncl).eq.2)) flg=0
c   check various possibilities of symmetry
      if (jj.gt.1) then
      if (ii.lt.jj) then
```

```
      aj=j-(jj-ii)*ncl
      q=0
      ai=i+(jj-ii-1)*ncl
      else
      aj=j-ncl
      ai=i-ncl
      q=1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      if (q.eq.0) then
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      else
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      endif
      else
c
      if ((jj.eq.1).and.(j.gt.1)) then
      if (j.le.ki) then
      aj=j-1
      ai=i-1
      q=0
      else
      aj=ki
      ai=i+(j-ki+1)
      q=1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      if (q.eq.0) then
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      endif
      else
c
      if ((j.eq.1).and.(ki.eq.2)) then
      aj=j
      ai=i-1
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
      endif
c
      if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower plate x-directed
c   and lower plate z-directed antenna conductor currents
      w1=wu(ip)/2.0
      h11=hu(ip)
      h21=hu(ip+ncl)
      w12=wu(j)/2.0
      w22=wu(j+1)/2.0
      h2=hu(j+1)
      hh=(jj-ii)*lcz
      d=(kj-ki)*lcx-w12+w1
      call ortot(d,hh,w1,h11,h21,w12,w22,h2,z12)
```

```
      z(dum2(j),dum2(i))=z12
      endif
c     endif
c
c
c     if ((jxl.eq.1).and.(iyu.eq.1)) then
c   investigate interaction between lower plate x-directed
c   and upper plate z-directed antenna conductor currents
      jj=int((j-1)/ncl)+1
      ii=int((i-3*m1-1)/ncl)+1
      ki=i-3*m1-(ii-1)*ncl
      kj=j-(jj-1)*ncl
      ip=i-(2*ii-1)*ncl-2*m1
      if ((dum(j).eq.2).or.(dum(j+1).eq.2)) flg=0
      if ((dum(i).eq.2).or.(dum(i+ncl).eq.2)) flg=0
c   check various possibilities of symmetry
      if ((jj.gt.1).and.(ii.gt.1)) then
      aj=j-ncl
      ai=i-ncl
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
c     if ((jj.eq.1).and.(j.gt.1)) then
      if (j.le.ki) then
      aj=j-1
      ai=i-1
      q=0
      else
      aj=ki
      ai=i+(j-ki+1)
      q=1
      endif
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      if (q.eq.0) then
      z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
      else
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      endif
      else
c     if ((kj.eq.1).and.(ki.eq.2)) then
      aj=j
      ai=i-1
      if ((dum2(aj).eq.(0)).or.(dum2(ai).eq.(0))) flg=0
      if ((dum(aj).eq.2).or.(dum(aj+1).eq.2)) flg=0
      if ((dum(ai).eq.2).or.(dum(ai+ncl).eq.2)) flg=0
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      else
      flg=0
      endif
      endif
      endif
c     if (flg.eq.0) then
c   no symmetry
c   calculate interaction between lower plate x-directed
c   and upper plate z-directed antenna conductor currents
      w1=wu(ip)/2.0
      h11=hu(ip)
```

```
      h21=hu(ip-ncl)
      w12=wu(j)/2.0
      w22=wu(j+1)/2.0
      h2=hu(j+1)
      hh=(jj-ii-ncw-1)*lcz-wf+h11
      d=(kj-ki)*lcx-w12+w1
      call ortot(d,hh,w1,h11,h21,w12,w22,h2,z12)
      z(dum2(j),dum2(i))=z12
      endif
c
      endif
c
ccc
      if ((jyl.eq.1).and.(ixl.eq.1)) then
c   use symmetry for interaction of upper plate z-directed
c   and lower plate x-directed currents (j>i)
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
      if ((jyl.eq.1).and.(ixu.eq.1)) then
c   use symmetry for interaction of lower plate z-directed
c   and upper plate x-directed currents (j>i)
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
      if ((jxu.eq.1).and.(iyl.eq.1)) then
c   use symmetry for interaction of upper plate x-directed
c   and lower plate z-directed currents
      jj=int((j-m1-1)/ncl)+1
      ii=int((i-2*m1-1)/ncl)+1
      ai=i+2*(ncw-ii)*ncl
      aj=j-(2*jj-1)*ncl
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      endif
c
      if ((jxu.eq.1).and.(iyu.eq.1)) then
c   use symmetry for interaction of upper plate x-directed
c   and upper plate z-directed currents
      jj=int((j-m1-1)/ncl)+1
      ii=int((i-3*m1-1)/ncl)+1
      ai=i-2*ii*ncl
      aj=j-(2*jj-1)*ncl
      z(dum2(j),dum2(i))=-z(dum2(aj),dum2(ai))
      endif
c
      if ((jyl.eq.1).and.(ixl.eq.1)) then
c   use symmetry for interaction of lower plate z-directed
c   and lower plate x-directed currents (j>i)
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
      if ((jyu.eq.1).and.(ixl.eq.1)) then
c   use symmetry for interaction of upper plate z-directed
c   and lower plate x-directed currents (j>i)
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
      if ((jyu.eq.1).and.(ixu.eq.1)) then
c   use symmetry for interaction of upper plate z-directed
c   and upper plate x-directed currents (j>i)
      z(dum2(j),dum2(i))=z(dum2(i),dum2(j))
      endif
c
      if ((jyu.eq.1).and.(iyu.eq.1)) then
c   use symmetry for interaction of upper plate z-directed
```

```
c    and upper plate z-directed currents
       jj=int((j-3*m1-1)/ncl)+1
       ii=int((i-3*m1-1)/ncl)+1
       aj=j-2*jj*ncl
       ai=i-2*ii*ncl
       z(dum2(j),dum2(i))=z(dum2(aj),dum2(ai))
       endif
c
    30 continue
       return
       end
```

```
      subroutine filcd(lcx,lcz,lx,ly,lz,imp,dum,dum2,mc1,
     +       ncl,ncw,md1,md2,nld,nwd,hghd,wdth,wf,elnumc,
     +       elnumd,wu,hu,xs0,wsc,lsc,n,mi,xd0,wq,hf,nf)
      real    w1,h1,d,hh,l,i1,i2,lcx,lcz,lx,ly,lz,wu(2000),
     +        wdth,wf,hghd,lsc,wsc,xs0,pjw,pjh,wq,
     +        hfj,h11,h21,hu(2000),pi,xd0,hf
      complex imp(n,n),z12,z,temp
      integer dum(2000),dum2(2000),mc1,ncl,ncw,md1,md2,nld,jxl,
     +        jxu,jyl,jyu,it,ih,iw,il,jj,aj,i,j,sym,
     +        jp,elnumc,elnumd,n,mi,nf,jfl,
     +        ip,ii,ai,nwd,nw
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filcd computes the conductor-dielectric                  c
c   interactions of the impedance matrix (submatrix C)                  c
c                                                                       c
c   called by: mom                                                      c
c                                                                       c
c   calls: zcdxx, zcdxy, zcdxz, zcdzx, zcdzy, zcdzz                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      pi=atan(1.0)*4.0
      nw=nwd/2
      sym=1
      do 30 j=1,4*mc1+nf
      do 30 i=1,3*md2
      if (dum2(j).eq.0) goto 30
c   initialize
      jxl=0
      jxu=0
      jyl=0
      jyu=0
      jfl=0
      it=0
      ih=0
      iw=0
      il=0
c   determine the location of the conductor current
      if ((j.ge.0).and.(j.le.mc1)) jxl=1
      if ((j.gt.mc1).and.(j.le.(2*mc1))) jxu=1
      if ((j.gt.(2*mc1)).and.(j.le.(3*mc1))) jyl=1
      if ((j.gt.(3*mc1)).and.(j.le.(4*mc1))) jyu=1
      if ((j.gt.(4*mc1+1)).and.(j.le.(4*mc1+nf))) jfl=1
c   determine the location of the dielectric current
      it=int((i-1)/md2)
      ih=int((i-it*md2-1)/md1)
      iw=int((i-it*md2-ih*md1-1)/nld)
      il=i-it*md2-ih*md1-iw*nld
      z12=0.0
c
      if (jxl.eq.1) then
c   calculate interaction between lower plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zcdxx,y,z resp.
      jj=int((j-1)/ncl)+1
      h1=hu(j+1)*2.0*pi
      w1=lcx*pi
      d=((il-0.50)*lx+xd0-(j-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-(jj-1)*lcz)*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdxx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
```

```
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      if (it.eq.0) call zcdxx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
      endif
c
      if (jxu.eq.1) then
c   investigate interaction between upper plate x-directed
c   conductor current and dielectric currents
      jj=int((j-mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl
      if ((iw+1).gt.nw) then
      ai=i-(2*(iw+1-nw)-1)*nld
      else
      ai=i+(2*(nw-(iw+1))+1)*nld
      endif
c
      if (sym.eq.0) then
c  no symmetry
c   calculate interaction between upper plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zcdxx,y,z resp.
      w1=lcx*pi
      h1=hu(jp+1)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(j-mc1-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-((jj)*lcz+wdth+wf-hu(jp+1)))*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdxx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      if (it.eq.0) call zcdxx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdxy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdxz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
      else
      if (it.eq.0) imp(i+elnumc,dum2(j))=imp(ai+elnumc,dum2(jp))
      if (it.eq.1) imp(i+elnumc,dum2(j))=imp(ai+elnumc,dum2(jp))
      if (it.eq.2) imp(i+elnumc,dum2(j))=-imp(ai+elnumc,dum2(jp))
      endif
      endif
c
      if (j.eq.(4*mc1+1)) then
c   calculate interaction between short-circuit or receiving
c   diode current and dielectric currents if dielectric
c   current is x,y,z directed call zcdzx,y,z resp.
      w1=wsc*2.0*pi
      h1=lsc*2.0*pi
      d=(il*lx-lx/2.0+xd0-wsc)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=((iw+1)*lz-lz/2.0-(wdth+wf/2.0)+lsc)*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
```

```
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-isc*2.0*pi
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
      endif
c
      if (jfl.eq.1) then
c   calculate interaction between microstripline
c   current and dielectric currents if dielectric
c   current is x,y,z directed call zcdzx,y,z resp.
      w1=wq*pi
      h1=hf*2.0*pi
      jj=j-4*mc1-1
      hfj=(jj-1)*hf
      d=(il*lx-lx/2.0+xd0-xs0)*2.0*pi
      hh=-((ih+1)*ly-ly/2.0)*2.0*pi
      l=((iw+1)*lz-lz/2.0-hfj)*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
      endif
c
      if (jyl.eq.1) then
c   calculate interaction between lower plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zcdzx,y,z resp.
      jj=int((j-2*mc1-1)/ncl)+1
      jp=j-2*mc1
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(jp-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=((iw+1)*lz-lz/2.0-(jj-1)*lcz)*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp+ncl)*2.0*pi
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
```

```
      endif
c
      if (jyu.eq.1) then
c   investigate interaction between upper plate x-directed
c   conductor current and dielectric currents
      jj=int((j-3*mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl-2*mc1
      aj=j-2*jj*ncl
      if ((iw+1).gt.nw) then
      ai=i-(2*(iw+1-nw)-1)*nld
      else
      ai=i+(2*(nw-(iw+1))+1)*nld
      endif
c
      if (sym.eq.0) then
c   no symmetry
c   calculate interaction between upper plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zcdzx,y,z resp.
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(j-3*mc1-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=((iw+1)*lz-lz/2.0-(jj*lcz+wdth+wf-hu(jp)))*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp-ncl)*2.0*pi
      if (it.eq.0) call zcdzx(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zcdzy(d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zcdzz(d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(i+elnumc,dum2(j))=z12
      else
      if (it.eq.0) imp(i+elnumc,dum2(j))=-imp(ai+elnumc,dum2(aj))
      if (it.eq.1) imp(i+elnumc,dum2(j))=-imp(ai+elnumc,dum2(aj))
      if (it.eq.2) imp(i+elnumc,dum2(j))=imp(ai+elnumc,dum2(aj))
      endif
      endif
c
   30 continue
      return
      end
```

```
      subroutine fildc(lcx,lcz,lx,ly,lz,imp,dum,dum2,mc1,
     +       ncl,ncw,md1,md2,nld,nwd,hghd,wdth,wf,elnumc,
     +       elnumd,wu,hu,xs0,wsc,lsc,n,mi,xd0,wq,hf,nf)
      real    w1,h1,d,hh,l,i1,i2,lcx,lcz,lx,ly,lz,wu(2000),xd0,
     +        wdth,wf,hghd,lsc,wsc,xs0,pjw,pjh,wq,hf,hfj,
     +        h11,h21,hu(2000),
     +        pi,klx,kly,klz
      complex imp(n,n),z12,z,temp,maxz
      integer dum(2000),dum2(2000),mc1,ncl,ncw,md1,md2,nld,jxl,
     +        jxu,jyl,jyu,it,ih,iw,il,jj,aj,i,j,nw,nwd,ai,ii,
     +        jp,elnumc,elnumd,n,mi,sym,jfl,nf
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filcd computes the dielectric-conductor              c
c   interactions of the impedance matrix (submatrix B)              c
c                                                                   c
c   called by: mom                                                  c
c                                                                   c
c   calls: zdcxx, zdcxy, zdcxz, zdczx, zdczy, zdczz                 c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      pi=atan(1.0)*4.0
      klx=2.0*pi*lx
      kly=2.0*pi*ly
      klz=2.0*pi*lz
      nw=nwd/2
      sym=1
c
      do 30 j=1,4*mc1+nf
      do 30 i=1,3*md2
      if (dum2(j).eq.0) goto 30
c   initialize
      jxl=0
      jxu=0
      jyl=0
      jyu=0
      jfl=0
      it=0
      ih=0
      iw=0
      il=0
c   determine the location of the conductor current
      if ((j.ge.0).and.(j.le.mc1)) jxl=1
      if ((j.gt.mc1).and.(j.le.(2*mc1))) jxu=1
      if ((j.gt.(2*mc1)).and.(j.le.(3*mc1))) jyl=1
      if ((j.gt.(3*mc1)).and.(j.le.(4*mc1))) jyu=1
      if ((j.gt.(4*mc1+1)).and.(j.le.(4*mc1+nf))) jfl=1
c   determine the location of the dielectric current
      it=int((i-1)/md2)
      ih=int((i-it*md2-1)/md1)
      iw=int((i-it*md2-ih*md1-1)/nld)
      il=i-it*md2-ih*md1-iw*nld
      z12=0.0
c
      if (jxl.eq.1) then
c   calculate interaction between lower plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zdcxx,y,z resp.
      jj=int((j-1)/ncl)+1
      h1=hu(j+1)*2.0*pi
      w1=lcx*pi
      d=((il-0.50)*lx+xd0-(j-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-(jj-1)*lcz)*2.0*pi
c
      if (abs(l).gt.(9.0*h1)) then
```

```
      if (it.eq.0) z12=-imp(i+elnumc,dum2(j))*klx
      if (it.eq.1) z12=-imp(i+elnumc,dum2(j))*kly
      if (it.eq.2) z12=-imp(i+elnumc,dum2(j))*klz
      else
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdcxx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdcxy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdcxz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      if (it.eq.0) call zdcxx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdcxy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdcxz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      endif
      imp(dum2(j),i+elnumc)=z12
      endif
c
      if (jxu.eq.1) then
c   investigate interaction between upper plate x-directed
c   conductor current and dielectric currents
      jj=int((j-mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl
      if ((iw+1).gt.nw) then
      ai=i-(2*(iw+1-nw)-1)*nld
      else
      ai=i+(2*(nw-(iw+1))+1)*nld
      endif
c
      if (sym.eq.0) then
c  no symmetry
c   calculate interaction between upper plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zcdxx,y,z resp.
      w1=lcx*pi
      h1=hu(jp+1)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(j-mc1-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-(jj*lcz+wdth+wf-hu(jp+1)))*2.0*pi
      if (abs(l).gt.(9.0*h1)) then
      if (it.eq.0) z12=-imp(i+elnumc,dum2(j))*klx
      if (it.eq.1) z12=-imp(i+elnumc,dum2(j))*kly
      if (it.eq.2) z12=-imp(i+elnumc,dum2(j))*klz
      else
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdcxx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdcxy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdcxz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      d=d-lcx*2.0*pi
      if (it.eq.0) call zdcxx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdcxy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdcxz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      endif
      else
      if (it.eq.0) z12=imp(dum2(jp),ai+elnumc)
      if (it.eq.1) z12=imp(dum2(jp),ai+elnumc)
      if (it.eq.2) z12=-imp(dum2(jp),ai+elnumc)
```

```
      endif
      imp(dum2(j),i+elnumc)=z12
      endif
c
      if (j.eq.(4*mc1+1)) then
c   calculate interaction between short-circuit or receiving
c   diode current and dielectric currents if dielectric
c   current is x,y,z directed call zdczx,y,z resp.
      w1=wsc*2.0*pi
      h1=lsc*2.0*pi
      d=(il*lx-lx/2.0+xd0-wsc)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-(wdth+wf/2.0)+lsc)*2.0*pi
      if (abs(d).gt.(15.0*w1)) then
      if (it.eq.0) z12=-imp(i+elnumc,dum2(j))*klx
      if (it.eq.1) z12=-imp(i+elnumc,dum2(j))*kly
      if (it.eq.2) z12=-imp(i+elnumc,dum2(j))*klz
      else
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      endif
      imp(dum2(j),i+elnumc)=z12
      endif
c
      if (jfl.eq.1) then
c   calculate interaction between microstripline
c   current and dielectric currents if dielectric
c   current is x,y,z directed call zdczx,y,z resp.
      w1=wq*pi
      h1=hf*2.0*pi
      jj=j-4*mc1-1
      hfj=(jj-1)*hf
      d=(il*lx-lx/2.0+xd0-xs0)*2.0*pi
      hh=-((ih+1)*ly-ly/2.0)*2.0*pi
      l=((iw+1)*lz-lz/2.0-hfj)*2.0*pi
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      imp(dum2(j),i+elnumc)=z12
      endif
c
      if (jyl.eq.1) then
c   calculate interaction between lower plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zdczx,y,z resp.
```

```
      jj=int((j-2*mc1-1)/ncl)+1
      jp=j-2*mc1
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(jp-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-(jj-1)*lcz)*2.0*pi
      if (abs(d).gt.(15.0*w1)) then
      if (it.eq.0) z12=-imp(i+elnumc,dum2(j))*klx
      if (it.eq.1) z12=-imp(i+elnumc,dum2(j))*kly
      if (it.eq.2) z12=-imp(i+elnumc,dum2(j))*klz
      else
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
      i2=0.0
      l=l-h1
      h1=hu(jp+ncl)*2.0*pi
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      endif
      imp(dum2(j),i+elnumc)=z12
      endif
c
      if (jyu.eq.1) then
c   investigate interaction between upper plate x-directed
c   conductor current and dielectric currents
      jj=int((j-3*mc1-1)/ncl)+1
      jp=j-(2*jj-1)*ncl-2*mc1
      aj=j-2*jj*ncl
      if ((iw+1).gt.nw) then
      ai=i-(2*(iw+1-nw)-1)*nld
      else
      ai=i+(2*(nw-(iw+1))+1)*nld
      endif
c
      if (sym.eq.0) then
c   no symmetry
c   calculate interaction between upper plate x-directed
c   conductor current and dielectric currents if dielectric
c   current is x,y,z directed call zdczx,y,z resp.
      w1=lcx*pi
      h1=hu(jp)*2.0*pi
      d=(il*lx-lx/2.0+xd0-(j-3*mc1-(jj-1)*ncl)*lcx+lcx/2.0)*2.0*pi
      hh=((ih+1)*ly-ly/2.0)*2.0*pi
      l=(((iw+1)*lz-lz/2.0)-((jj)*lcz+wdth+wf-hu(jp)))*2.0*pi
      if (abs(d).gt.(15.0*w1)) then
      if (it.eq.0) z12=-imp(i+elnumc,dum2(j))*klx
      if (it.eq.1) z12=-imp(i+elnumc,dum2(j))*kly
      if (it.eq.2) z12=-imp(i+elnumc,dum2(j))*klz
      else
      i1=0.0
      i2=1.0
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      i1=1.0
```

```
      i2=0.0
      l=l-hl
      h1=hu(jp-ncl)*2.0*pi
      if (it.eq.0) call zdczx(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.1) call zdczy(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      if (it.eq.2) call zdczz(klx,kly,klz,d,hh,l,w1,h1,i1,i2,z)
      z12=z12+z
      endif
      else
      if (it.eq.0) z12=-imp(dum2(aj),ai+elnumc)
      if (it.eq.1) z12=-imp(dum2(aj),ai+elnumc)
      if (it.eq.2) z12=imp(dum2(aj),ai+elnumc)
      endif
      imp(dum2(j),i+elnumc)=z12
      endif
c
   30 continue
      return
      end
```

```
      subroutine fildd(lx,ly,lz,imp,nl,nh,
     +                 nw,m1,m2,epsr,freq,elnumc,n)
      real    pi,lx,ly,lz,d,hh,l,epsr,freq,
     +        klx,kly,klz,mult
      complex imp(n,n),z12,p1
      integer i,j,nl,nh,nw,m1,m2,jt,it,jl,il,jw,iw,sym,ok,np,
     +        is,js,elnumc,n,jh,ih
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine filcd computes the dielectric-dielectric              c
c   interactions of the impedance matrix (submatrix D)               c
c                                                                    c
c   called by: mom                                                   c
c                                                                    c
c   calls: zxx, zxy, zxz, zyy, zyz, zzz, symxx, symxy, symxz,        c
c          symyy, symyz, symzz                                       c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      pi=atan(1.0)*4.0
      klx=2.0*pi*lx
      kly=2.0*pi*ly
      klz=2.0*pi*lz
      p1=(0.0,1.0)
      np=elnumc
      sym=1
c
      do 10 j=1,3*m2
      do 10 i=1,3*m2
c   initialize
      ok=0
      mult=1.0
c   check obvious symmetry
      if (j.gt.i) then
      imp(j+np,i+np)=imp(i+np,j+np)
      else
c   determine locations of the currents i, j
      jt=int((j-1)/m2)
      jh=int((j-jt*m2-1)/m1)
      jw=int((j-jt*m2-jh*m1-1)/nl)
      jl=j-jt*m2-jh*m1-jw*nl
      it=int((i-1)/m2)
      ih=int((i-it*m2-1)/m1)
      iw=int((i-it*m2-ih*m1-1)/nl)
      il=i-it*m2-ih*m1-iw*nl
c   determine |r-r'|
      d=(jl-il)*klx
      hh=(ih-jh)*kly
      l=(jw-iw)*klz
c
      if ((jt.eq.0).and.(it.eq.0)) then
c   both currents in x-direction
      if (sym.eq.1) then
c   check symmetry
      call symxx(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
      endif
c   if no symmetry calculate interaction
      if (ok.eq.0) call zxx(klx,kly,klz,d,hh,l,z12)
      endif
c
      if ((jt.eq.0).and.(it.eq.1)) then
c   this interaction is zero for one y segment
c      if (sym.eq.1) then
c      call symxy(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
c      endif
c      if (ok.eq.0) call zxy(klx,kly,klz,d,hh,l,z12)
      z12=0.0
```

```
      endif
c
      if ((jt.eq.0).and.(it.eq.2)) then
c   currents in x and z direction
      if (sym.eq.1) then
c   check symmetry
      call symxz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
      endif
      if (ok.eq.0) then
      if ((abs(d).lt.klx).or.(abs(l).lt.klz)) then
      z12=0.0
      else
c   if no symmetry calculate interaction
      call zxz(klx,kly,klz,d,hh,l,z12)
      endif
      endif
      endif
c
      if ((jt.eq.1).and.(it.eq.1)) then
c   both currents in y-direction
      if (sym.eq.1) then
c   check symmetry
      call symyy(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
      endif
c   if no symmetry calculate interaction
      if (ok.eq.0) call zyy(klx,kly,klz,d,hh,l,z12)
      endif
c
      if ((jt.eq.1).and.(it.eq.2)) then
c      if (sym.eq.1) then
c      call symyz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
c      endif
c      if (ok.eq.0) call zyz(klx,kly,klz,d,hh,l,z12)
      z12=0.0
      endif
c
      if ((jt.eq.2).and.(it.eq.2)) then
c   both currents in z-direction
      if (sym.eq.1) then
c   check symmetry
      call symzz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,js,is,ok,mult)
      endif
c   if no symmetry calculate interaction
      if (ok.eq.0) call zzz(klx,kly,klz,d,hh,l,z12)
      endif
c
      if (ok.eq.1) then
c   use symmetry data to get the interaction
      imp(j+np,i+np)=mult*imp(js+np,is+np)
      else
      imp(j+np,i+np)=z12
      if (j.eq.i) then
c   add the self term from the field equality equation
      imp(j+np,i+np)=imp(j+np,i+np)+
     +                pi*120.0*pi/(epsr-1.0)
      endif
      endif
      endif
10  continue
c
      do 20 j=1,3*m2
      do 20 i=1,3*m2
c   multiply with the definition constants
      if ((i.ge.1).and.(i.le.m2)) then
      imp(j+np,i+np)=imp(j+np,i+np)/(kly*klz)
      endif
      if ((i.gt.m2).and.(i.le.(2*m2))) then
```

```
      imp(j+np,i+np)=imp(j+np,i+np)/(klx*klz)
      endif
      if ((i.gt.(2*m2)).and.(i.le.(3*m2))) then
      imp(j+np,i+np)=imp(j+np,i+np)/(klx*kly)
      endif
20    continue
      return
      end
```

```
      subroutine cgrad(imp,vlt,n,nu,cr)
      real   x1,x2,x3,tol,bk,alpha,c
      complex   imp(n,n),cr(n),vlt(n),
     +           p(n),r(n),temp(n)
      integer i,j,n,nu
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine cgrad solves nu by nu MoM matrix equation          c
c                                                                 c
c   called by: mom                                                c
c                                                                 c
c   calls: atrcgp, adot, uap, uapr                                c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   initialize the error and solution vectors
      c=1.e-4
      do 10 j=1,nu
      r(j)=vlt(j)
      cr(j)=0.0
   10 continue
c   calculate initial error norm
      call adot(n,nu,r,r,x1)
c   tolerance=c* initial error norm
      tol=c*sqrt(x1)
      write(6,*) 'initial error=',sqrt(x1)
c   calculate transpose conjugate of A times r ( =p )
      call atrcgp(n,nu,imp,r,p)
c   iterate
      do 30 i=1,2*n
c   calculate transpose conjugate of A times r ( =temp )
      call atrcgp(n,nu,imp,r,temp)
c   calculate norm of temp = x2
      call adot(n,nu,temp,temp,x2)
c
CDIR@ NOVECTOR
c
c   calculate A times p
      do 101 j2=1,nu
      temp(j2)=0.0
  101 continue
      do 301 j2=1,nu
CDIR@ VECTOR
      do 201 i2=1,nu
      temp(j2)=temp(j2)+imp(j2,i2)*p(i2)
  201 continue
  301 continue
c
c   calculate norm of temp = x3
      call adot(n,nu,temp,temp,x3)
      alpha=x2/x3
      call uap(n,nu,cr,alpha,p)
      alpha=-alpha
c   calculate new r (error) vector
      call uap(n,nu,r,alpha,temp)
c   check error norm against tolerance
      call adot(n,nu,r,r,x1)
      if (sqrt(x1).lt.tol) then
      itno=i
      goto 60
      endif
c
c   calculate transpose conjugate of A times r ( =temp )
      call atrcgp(n,nu,imp,r,temp)
c   calculate norm of temp = x3
      call adot(n,nu,temp,temp,x3)
      bk=x3/x2
c   calculate new p vector
```

```
      call uapr(n,nu,temp,p,bk)
   30 continue
      write(6,*) 2*n,'iterations without result'
      goto 62
   60 write(6,*) ' no of iterations = ',itno
   62 continue
      write(6,*) 'error now = ',sqrt(xi)
      return
      end
c
c
      subroutine atrcgp(n,nu,im,x,y)
      complex im(n,n),x(n),y(n)
      integer n,i,j,nu
c
c   calculates the multiplication of transpose conjugate of A (nu by nu)
c   with x, puts result into y vector
c
CDIR@ NOVECTOR
      do 11 i=1,nu
      y(i)=0.0
CDIR@ VECTOR
      do 10 j=1,nu
      y(i)=y(i)+conjg(im(j,i))*x(j)
  10  continue
  11  continue
      return
      end
c
c
      subroutine uapr(n,nu,temp,p,bk)
      real  bk
      complex   temp(n),p(n)
      integer n,i,nu
c
c   calculates p=temp+bk*p , bk is a constant
c
      do 10 i=1,nu
      p(i)=temp(i)+bk*p(i)
   10 continue
      return
      end
c
c
      subroutine adot (n,nu,r,s,t)
      real t
      complex r(n),s(n)
c
c   calculates the dot multiplication of r and s
c
      integer n,i,nu
      t=0.0
      do 10 i=1,nu
      t=t+conjg(r(i))*s(i)
   10 continue
      return
      end
c
c
      subroutine uap(n,nu,u,a,p)
      real a
      complex u(n),p(n)
      integer n,i,nu
c
c   calculates u_new = u_old + a * p , a is a constant
c
      do 10 i=1,nu
      u(i)=u(i)+a*p(i)
   10 continue
      return
      end
```

```
      subroutine partot(d,hh,w1,h11,h21,w2,h12,h22,z12)
      real d,hh,w1,h11,h21,w2,h12,h22,i11,i21,i12,i22
      complex z,z12
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine partot calculates the mutual impedance                  c
c   between two coplanar parallel surface dipoles i and j              c
c   Dipole i extends from monopole i1 to i2 whereas                    c
c   Dipole j extends from monopole j1 to j2                            c
c                                                                      c
c   called by: filcc                                                   c
c                                                                      c
c   calls: par                                                         c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      i11=0.0
      i21=1.0
      i12=0.0
      i22=1.0
c   calculate the first monopole interaction i1-j1
      call par(i11,i21,i12,i22,d,hh,w1,h11,w2,h12,z)
      z12=z
      i11=1.0
      i21=0.0
      hh=hh-h11
c   calculate the second monopole interaction i2-j1
      call par(i11,i21,i12,i22,d,hh,w1,h21,w2,h12,z)
      z12=z12+z
      i12=1.0
      i22=0.0
      hh=hh+h12
c   calculate the third monopole interaction i2-j2
      call par(i11,i21,i12,i22,d,hh,w1,h21,w2,h22,z)
      z12=z12+z
      i11=0.0
      i21=1.0
      hh=hh+h11
c   calculate the fourth monopole interaction i1-j2
      call par(i11,i21,i12,i22,d,hh,w1,h11,w2,h22,z)
      z12=z12+z
      return
      end
```

```
      subroutine parftot(d,hh,l,w1,h11,h21,w2,h12,h22,z12)
      real d,hh,w1,h11,h21,w2,h12,h22,i11,i21,i12,i22,l
      complex z,z12
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine partot calculates the mutual impedance                  c
c   between two non-planar parallel surface dipoles i and j            c
c   Dipole i extends from monopole i1 to i2 whereas                    c
c   Dipole j extends from monopole j1 to j2                            c
c                                                                      c
c   called by: filcc                                                   c
c                                                                      c
c   calls: parf
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      i11=0.0
      i21=1.0
      i12=0.0
      i22=1.0
c   calculate the first monopole interaction i1-j1
      call parf(i11,i21,i12,i22,d,hh,l,w1,h11,w2,h12,z)
      z12=z
      i11=1.0
      i21=0.0
      hh=hh-h11
c   calculate the second monopole interaction i2-j1
      call parf(i11,i21,i12,i22,d,hh,l,w1,h21,w2,h12,z)
      z12=z12+z
      i12=1.0
      i22=0.0
      hh=hh+h12
c   calculate the third monopole interaction i2-j2
      call parf(i11,i21,i12,i22,d,hh,l,w1,h21,w2,h22,z)
      z12=z12+z
      i11=0.0
      i21=1.0
      hh=hh+h11
c   calculate the fourth monopole interaction i1-j2
      call parf(i11,i21,i12,i22,d,hh,l,w1,h11,w2,h22,z)
      z12=z12+z
      return
      end
```

```
      subroutine ortot(d,hh,w1,h11,h21,w12,w22,h2,z12)
      real d,hh,w1,h11,h21,w12,w22,h2,i11,i21,i12,i22
      complex z,z12
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine ortot calculates the mutual impedance                  c
c   between two coplanar perpendicular surface dipoles i and j        c
c   Dipole i extends from monopole iz1 to iz2 whereas                 c
c   Dipole j extends from monopole jx1 to jx2                         c
c                                                                     c
c   called by: filcc                                                  c
c                                                                     c
c   calls: orthog                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      i11=0.0
      i21=1.0
      i12=0.0
      i22=1.0
c   calculate the first monopole interaction iz1-jx1
      call orthog(i11,i21,i12,i22,d,hh,w1,h11,w12,h2,z)
      z12=z
      i11=1.0
      i21=0.0
      hh=hh-h11
c   calculate the second monopole interaction iz2-jx1
      call orthog(i11,i21,i12,i22,d,hh,w1,h21,w12,h2,z)
      z12=z12+z
      i12=1.0
      i22=0.0
      d=d+w12+w22
c   calculate the third monopole interaction iz2-jx2
      call orthog(i11,i21,i12,i22,d,hh,w1,h21,w22,h2,z)
      z12=z12+z
      i11=0.0
      i21=1.0
      hh=hh+h11
c   calculate the fourth monopole interaction iz1-jx2
      call orthog(i11,i21,i12,i22,d,hh,w1,h11,w22,h2,z)
      z12=z12+z
      return
      end
```

```
      subroutine orftot(d,hh,l,w1,h11,h21,w12,w22,h2,z12)
      real d,hh,w1,h11,h21,w12,w22,h2,i11,i21,i12,i22,l
      complex z,z12
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine orftot calculates the mutual impedance              c
c   between two non-planar perpendicular surface dipoles i and j   c
c   Dipole i extends from monopole iz1 to iz2 whereas              c
c   Dipole j extends from monopole jx1 to jx2                      c
c                                                                  c
c   called by: filcc                                               c
c                                                                  c
c   calls: orf                                                     c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      i11=0.0
      i21=1.0
      i12=0.0
      i22=1.0
c   calculate the first monopole interaction iz1-jx1
      call orf(i11,i21,i12,i22,d,hh,l,w1,h11,w12,h2,z)
      z12=z
      i11=1.0
      i21=0.0
      hh=hh-h11
c   calculate the second monopole interaction iz2-jx1
      call orf(i11,i21,i12,i22,d,hh,l,w1,h21,w12,h2,z)
      z12=z12+z
      i12=1.0
      i22=0.0
      d=d+w12+w22
c   calculate the third monopole interaction iz2-jx2
      call orf(i11,i21,i12,i22,d,hh,l,w1,h21,w22,h2,z)
      z12=z12+z
      i11=0.0
      i21=1.0
      hh=hh+h11
c   calculate the fourth monopole interaction iz1-jx2
      call orf(i11,i21,i12,i22,d,hh,l,w1,h11,w22,h2,z)
      z12=z12+z
      return
      end
```

```
      subroutine par(i11,i21,i12,i22,d,hh,w1,h1,w2,h2,z12)
      real     pi,i11,i21,i12,i22,d,hh,h1,w1,kd,kw1,kh1,khh,ab(3),
     +         gamma,x1,x2,y1,y2,s1,s2,a(2),h2,w2,kh2,kw2,
     +         z1(3),z2(3),b(2)
      complex   z12,z,pi,zd
      integer       i,flag,j,ng
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine par calculates the mutual impedance                 c
c   between two coplanar parallel surface monopoles i and j        c
c                                                                  c
c   called by: partot                                              c
c                                                                  c
c   calls: gq1d, fun1, ff1, ff2, fun2, ff3, ff4, ff, zz,           c
c          gaus2, gaus4, gaus6, gaus24                             c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   calculate the constants and initialize
c
      p1=(0.0,1.0)
      pi=atan(1.0)*4.0
      kd=2.0*pi*d
      kw1=2.0*pi*w1
      kw2=2.0*pi*w2
      kh1=2.0*pi*h1
      kh2=2.0*pi*h2
      khh=2.0*pi*hh
      a(1)=-i11+i21*cos(kh1)
      a(2)=i11*cos(kh1)-i21
      b(1)=khh-kh1
      b(2)=khh
      ab(1)=kw1+kw2-kd
      ab(2)=2.0*kw2
      if (w1.lt.w2) then
      ab(2)=2.0*kw1
      endif
      ab(3)=kw1+kw2+kd
      x1=0.0
      x2=kh2
      y1=abs(kw1-kw2)
      y2=kw1+kw2
      s1=-y2
      s2=y2
      z1(1)=-y2
      z2(1)=-y1
      z1(2)=-y1
      z2(2)=y1
      z1(3)=y1
      z2(3)=y2
      gamma=7.50/(kw1*kw2*sin(kh1)*sin(kh2))
      z12=0.0
      ng=24
      if (abs(d).gt.(2.0*(w1+w2)/2.0)) ng=6
      if (abs(d).gt.(4.0*(w1+w2)/2.0)) ng=4
      if (abs(d).gt.(6.0*(w1+w2)/2.0)) ng=2
c
c   calculate the mutual impedance
c
      do 10 i=1,2
      flag=1
      call gq1d(khh,kh1,kh2,kd,z,x1,x2,ab,z1,z2,
     +          kw1,kw2,i12,i22,b(i),flag,ng)
      z12=z12+z*a(i)
      flag=2
      zd=0.0
      do 17 j=1,3
```

```
      if (z1(j).eq.z2(j)) goto 17
      call gq1d(khh,kh1,kh2,kd,z,z1(j),z2(j),ab,z1,z2,
     +          kw1,kw2,i12,i22,b(i),flag,ng)
      z12=z12+z*a(i)
   17 continue
   10 continue
      z12=z12*gamma*(-p1)
      return
      end
c
c   gq1d - calculates integral in one dimension
c
      subroutine gq1d(khh,kh1,kh2,kd,z,x1,x2,ab,z1,z2,
     +                kw1,kw2,i12,i22,b,flag,ngaus)
      real    kd,x1,x2,kw1,kw2,xm,xr,p1,p2,x(24),w(24),dx,i12,
     +        khh,i22,kh1,kh2,ab(3),z1(3),z2(3),b,dist,pi
      complex z,r1,r2
      integer j,flag,ngaus
c
      if (ngaus.eq.24) call gaus24(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      if (ngaus.eq.4) call gaus4(x,w)
      if (ngaus.eq.2) call gaus2(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 j=1,ngaus
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      if (flag.eq.1) then
      call fun1(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,i12,i22,b,p1,r1)
      call fun1(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,i12,i22,b,p2,r2)
      else
      call fun2(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,i12,i22,b,p1,r1)
      call fun2(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,i12,i22,b,p2,r2)
      endif
      z=z+w(j)*(r1+r2)
   10 continue
      z=xr*z
      return
      end
c
c   fun1 calculates the integrand of first integral
c
      subroutine  fun1(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +                 i12,i22,b,x,z12)
      real    kd,b,kw1,kw2,i12,i22,x,x1,x2,ab(3),z1(3),z2(3),
     +        kh1,kh2,khh
      complex   z,p1,z12
      integer j
c
      z12=0.0
      p1=(0.0,1.0)
      call ff1(kd,b,z1(1),z2(1),x,z,kh2,i12,i22)
      z12=z12+z
      call ff1(kd,b,z1(3),z2(3),x,z,kh2,i12,i22)
      z12=z12-z
      do 10 j=1,3
      call ff2(kd,b,z1(j),z2(j),x,z,kh2,i12,i22)
      z12=z12+ab(j)*z
   10 continue
      z12=z12*p1
      return
      end
c
      subroutine ff1(kd,b,y1,y2,x,z,kh2,i12,i22)
```

```
      real  kd,b,y1,y2,x,kh2,i12,i22,c1,c2,c3
      complex  z,p1
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+y2)**2+(b+x)**2)
      c2=sqrt((kd+y1)**2+(b+x)**2)
      call ff(x,kh2,i12,i22,c3)
      z=(cexp(p1*c1)-cexp(p1*c2))*c3
      return
      end
c
      subroutine ff2(kd,b,y1,y2,x,z,kh2,i12,i22)
      real  kd,b,y1,y2,x,kh2,i12,i22,c1,c2,c3
      complex z,p1,c4,c5
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+y2)**2+(b+x)**2)
      c2=sqrt((kd+y1)**2+(b+x)**2)
      call ff(x,kh2,i12,i22,c3)
      if (c1.eq.(0.0)) then
      c4=1.0
      else
      c4=cexp(p1*c1)*(kd+y2)/c1
      endif
      if (c2.eq.(0.0)) then
      c5=1.0
      else
      c5=cexp(p1*c2)*(kd+y1)/c2
      endif
      z=(c4-c5)*p1*c3
      return
      end
c
c   fun2 calculates the integrand of 2nd integral
c
       subroutine  fun2(khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
      +                 i12,i22,b,x,z12)
       real    kd,b,kw1,kw2,i12,i22,x,x1,x2,ab(3),z1(3),z2(3),
      +        kh1,kh2,khh
       complex   z,p1,z12
c
       z12=0.0
       p1=(0.0,1.0)
       call ff3(kd,b,kh2,x,z,i12,i22)
       z12=z12+z
       call ff4(kd,b,kh2,x,z,i12,i22)
       z12=z12-z
       if (x.lt.z1(2)) then
       z12=z12*ab(1)
       endif
       if ((x.ge.z1(2)).and.(x.lt.z2(2))) then
       z12=z12*ab(2)
       endif
       if (x.ge.z2(2)) then
       z12=z12*ab(3)
       endif
       z12=z12*p1
       return
       end
c
       subroutine ff3(kd,b,kh2,x,z,i12,i22)
       real    kd,b,kh2,x,i12,i22,c1,c2,c3,c4
       complex   z,p1,c5,c6
c
       p1=(0.0,-1.0)
       c1=sqrt((kd+x)**2+(b+kh2)**2)
       c2=sqrt((kd+x)**2+b**2)
       call ff(kh2,kh2,i12,i22,c3)
```

```
      call ff((0.0),kh2,i12,i22,c4)
      if (c1.eq.(0.0)) then
      c5=c3
      else
      c5=cexp(p1*c1)*(b+kh2)*c3/c1
      endif
      if (c2.eq.(0.0)) then
      c6=c4
      else
      c6=cexp(p1*c2)*b*c4/c2
      endif
      z=(c5-c6)*p1
      return
      end
c
      subroutine ff4(kd,b,kh2,x,z,i12,i22)
      real    kd,b,kh2,x,i12,i22,c1,c2,c3,c4
      complex   z,p1
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+x)**2+(b+kh2)**2)
      c2=sqrt((kd+x)**2+b**2)
      call zz(kh2,kh2,i12,i22,c3)
      call zz((0.0),kh2,i12,i22,c4)
      z=cexp(p1*c1)*c3-cexp(p1*c2)*c4
      return
      end
c
      subroutine ff(x,kw,i12,i22,y)
      real    x,kw,i12,i22,y
      y=i12*sin(kw-x)+i22*sin(x)
      return
      end
c
      subroutine zz(x,kw,i12,i22,y)
      real   x,kw,i12,i22,y
      y=-i12*cos(kw-x)+i22*cos(x)
      return
      end
```

```
      subroutine parf(i11,i21,i12,i22,d,hh,t,w1,h1,w2,h2,z12)
      real  pi,i11,i21,i12,i22,d,hh,h1,w1,kd,kw1,kh1,khh,ab(3),
     +         gamma,x1,x2,y1,y2,s1,s2,a(2),h2,w2,kh2,kw2,
     +         z1(3),z2(3),b(2),t,kt,dist
      complex   z12,z,pi,zd
      integer       i,typ,flag,j,flag2
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine parf calculates the mutual impedance                c
c   between two non-planar parallel surface monopoles i and j      c
c                                                                  c
c   called by: parftot                                             c
c                                                                  c
c   calls: gq2df, gq1df, fun1f, funf3, ff1f, ff2f, fun2f,          c
c          ff3f, ff4f, fff, zzf, gaus6                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   calculate the constants and initialize
c
      p1=(0.0,1.0)
      pi=atan(1.0)*4.0
      kd=2.0*pi*d
      kw1=2.0*pi*w1
      kw2=2.0*pi*w2
      kh1=2.0*pi*h1
      kh2=2.0*pi*h2
      khh=2.0*pi*hh
      kt=2.0*pi*t
      a(1)=-i11+i21*cos(kh1)
      a(2)=i11*cos(kh1)-i21
      b(1)=khh-kh1
      b(2)=khh
      ab(1)=kw1+kw2-kd
      ab(2)=2.0*kw2
      if (w1.lt.w2) then
      ab(2)=2.0*kw1
      endif
      ab(3)=kw1+kw2+kd
      x1=0.0
      x2=kh2
      y1=abs(kw1-kw2)
      y2=kw1+kw2
      s1=-y2
      s2=y2
      z1(1)=-y2
      z2(1)=-y1
      z1(2)=-y1
      z2(2)=y1
      z1(3)=y1
      z2(3)=y2
      gamma=7.50/(kw1*kw2*sin(kh1)*sin(kh2))
      z12=0.0
      flag2=0
      dist=sqrt(d**2+hh**2)
      if (dist.ge.(100.0*abs(t))) flag2=1
c
c   calculate the mutual impedance
c
      do 10 i=1,2
      if (flag2.eq.1) then
      flag=1
      call gq1df(kt,khh,kh1,kh2,kd,z,x1,x2,ab,z1,z2,
     +          kw1,kw2,i12,i22,b(i),flag)
      z12=z12+z*a(i)
      flag=2
      zd=0.0
```

```
      endif
      do 17 j=1,3
      if (z1(j).eq.z2(j)) goto 17
      if (flag2.eq.1) then
      call gq1df(kt,khh,kh1,kh2,kd,z,z1(j),z2(j),ab,z1,z2,
     +          kw1,kw2,i12,i22,b(i),flag)
      else
      call gq2df(kt,khh,kh1,kh2,kd,z,z1(j),z2(j),x1,x2,ab,z1,z2,
     +          kw1,kw2,i12,i22,b(i),flag)
      endif
      z12=z12+z*a(i)
   17 continue
   10 continue
      z12=z12*gamma*(-p1)
      return
      end
c
c   g2qdf - calculates integral in two dimensions
c
      subroutine gq2df(kt,khh,kh1,kh2,kd,z,x1,x2,y1,y2,ab,z1,z2,
     +                 kw1,kw2,i12,i22,b,flag)
      real kd,x1,x2,kw1,kw2,xm,xr,p1,p2,x(24),w(24),dx,i12,
     +        khh,i22,kh1,kh2,ab(3),z1(3),z2(3),b,dist,pi,y1,y2,
     +        ym,yr,dy,q1,q2,kt
      complex z,r1,r2,r3,r4
      integer j,flag,ngaus
c
      pi=4.0*atan(1.0)
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      z=0.0
      do 10 j=1,ngaus
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      do 11 i=1,ngaus
      dy=yr*x(i)
      q1=ym+dy
      q2=ym-dy
      call funf3(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,q1,p1,r1)
      call funf3(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,q1,p2,r2)
      call funf3(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,q2,p1,r3)
      call funf3(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,q2,p2,r4)
      z=z+w(j)*w(i)*(r1+r2+r3+r4)
   11 continue
   10 continue
      z=xr*yr*z
      return
      end
c
c   g1qdf - calculates integral in one dimension
c
      subroutine gq1df(kt,khh,kh1,kh2,kd,z,x1,x2,ab,z1,z2,
     +                 kw1,kw2,i12,i22,b,flag)
      real kd,x1,x2,kw1,kw2,xm,xr,p1,p2,x(24),w(24),dx,i12,
     +        khh,i22,kh1,kh2,ab(3),z1(3),z2(3),b,dist,pi,kt
```

```
      complex z,r1,r2
      integer j,flag,ngaus
c
      pi=4.0*atan(1.0)
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 j=1,ngaus
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      if (flag.eq.1) then
      call fun1f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,p1,r1)
      call fun1f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,p2,r2)
      else
      call fun2f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,p1,r1)
      call fun2f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +           i12,i22,b,p2,r2)
      endif
      z=z+w(j)*(r1+r2)
   10 continue
      z=xr*z
      return
      end
c
c   fun1 calculates the integrand of first integral
c
      subroutine  fun1f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +                  i12,i22,b,x,z12)
      real kd,b,kw1,kw2,i12,i22,x,x1,x2,ab(3),z1(3),z2(3),
     +        kh1,kh2,khh,kt
      complex z,p1,z12
      integer j
c
      z12=0.0
      p1=(0.0,1.0)
      call ff1f(kt,kd,b,z1(1),z2(1),x,z,kh2,i12,i22)
      z12=z12+z
      call ff1f(kt,kd,b,z1(3),z2(3),x,z,kh2,i12,i22)
      z12=z12-z
      do 10 j=1,3
      call ff2f(kt,kd,b,z1(j),z2(j),x,z,kh2,i12,i22)
      z12=z12+ab(j)*z
   10 continue
      z12=z12*p1
      return
      end
c
c   funf3 calculates the integrand
c
      subroutine  funf3(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +                  i12,i22,b,y,x,z12)
      real kd,b,kw1,kw2,i12,i22,x,x1,x2,ab(3),z1(3),z2(3),
     +        kh1,kh2,khh,kt,y,c1,c2
      complex z,p1,z12
      integer j
c
      z12=0.0
      p1=(0.0,1.0)
      call fff(y,kh2,i12,i22,c2)
      c1=sqrt((kd+x)**2+(b+y)**2+kt**2)
      z12=cexp(-p1*c1)*c2/c1
```

```
      if (x.lt.z1(2)) then
      z12=z12*(x+kw1+kw2)
      endif
      if ((x.ge.z1(2)).and.(x.lt.z2(2))) then
      z12=z12*ab(2)
      endif
      if (x.ge.z2(2)) then
      z12=z12*(-x+kw1+kw2)
      endif
      return
      end
c
      subroutine ff1f(kt,kd,b,y1,y2,x,z,kh2,i12,i22)
      real kd,b,y1,y2,x,kh2,i12,i22,c1,c2,c3,kt
      complex z,p1
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+y2)**2+(b+x)**2+kt**2)
      c2=sqrt((kd+y1)**2+(b+x)**2+kt**2)
      call fff(x,kh2,i12,i22,c3)
      z=(cexp(p1*c1)-cexp(p1*c2))*c3
      return
      end
c
      subroutine ff2f(kt,kd,b,y1,y2,x,z,kh2,i12,i22)
      real kd,b,y1,y2,x,kh2,i12,i22,c1,c2,c3,kt
      complex z,p1,c4,c5
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+y2)**2+(b+x)**2+kt**2)
      c2=sqrt((kd+y1)**2+(b+x)**2+kt**2)
      call fff(x,kh2,i12,i22,c3)
      if (c1.eq.(0.0)) then
      c4=1.0
      else
      c4=cexp(p1*c1)*(kd+y2)/c1
      endif
      if (c2.eq.(0.0)) then
      c5=1.0
      else
      c5=cexp(p1*c2)*(kd+y1)/c2
      endif
      z=(c4-c5)*p1*c3
      return
      end
c
c   fun2f calculates the integrand of 2nd integral
c
      subroutine  fun2f(kt,khh,kh1,kh2,kd,x1,x2,ab,z1,z2,kw1,kw2,
     +                  i12,i22,b,x,z12)
      real kd,b,kw1,kw2,i12,i22,x,x1,x2,ab(3),z1(3),z2(3),
     +        kh1,kh2,khh,kt
      complex z,p1,z12
c
      z12=0.0
      p1=(0.0,1.0)
      call ff3f(kt,kd,b,kh2,x,z,i12,i22)
      z12=z12+z
      call ff4f(kt,kd,b,kh2,x,z,i12,i22)
      z12=z12-z
      if (x.lt.z1(2)) then
      z12=z12*ab(1)
      endif
      if ((x.ge.z1(2)).and.(x.lt.z2(2))) then
      z12=z12*ab(2)
      endif
      if (x.ge.z2(2)) then
      z12=z12*ab(3)
```

```
      endif
      z12=z12*p1
      return
      end
c
      subroutine ff3f(kt,kd,b,kh2,x,z,i12,i22)
      real kd,b,kh2,x,i12,i22,c1,c2,c3,c4,kt
      complex z,p1,c5,c6
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+x)**2+(b+kh2)**2+kt**2)
      c2=sqrt((kd+x)**2+b**2+kt**2)
      call fff(kh2,kh2,i12,i22,c3)
      call fff((0.0),kh2,i12,i22,c4)
      if (c1.eq.(0.0)) then
      c5=c3
      else
      c5=cexp(p1*c1)*(b+kh2)*c3/c1
      endif
      if (c2.eq.(0.0)) then
      c6=c4
      else
      c6=cexp(p1*c2)*b*c4/c2
      endif
      z=(c5-c6)*p1
      return
      end
c
      subroutine ff4f(kt,kd,b,kh2,x,z,i12,i22)
      real kd,b,kh2,x,i12,i22,c1,c2,c3,c4,kt
      complex z,p1
c
      p1=(0.0,-1.0)
      c1=sqrt((kd+x)**2+(b+kh2)**2+kt**2)
      c2=sqrt((kd+x)**2+b**2+kt**2)
      call zzf(kh2,kh2,i12,i22,c3)
      call zzf((0.0),kh2,i12,i22,c4)
      z=cexp(p1*c1)*c3-cexp(p1*c2)*c4
      return
      end
c
      subroutine fff(x,kw,i12,i22,y)
      real x,kw,i12,i22,y
      y=i12*sin(kw-x)+i22*sin(x)
      return
      end
c
      subroutine zzf(x,kw,i12,i22,y)
      real x,kw,i12,i22,y
      y=-i12*cos(kw-x)+i22*cos(x)
      return
      end
```

```
      subroutine orthog(i11,i21,i12,i22,d,hh,w1,h1,w2,h2,z12)
      real     pi,i11,i21,i12,i22,d,hh,h1,w1,kd,kw1,kh1,khh,
     +         a(2),h2,w2,kw2,kh2,gamma,x1(3),x2(3),spt,c1
      complex     z12,z,cj
      integer       i,j,typ,flag,ng
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine orthog calculates the mutual impedance              c
c   between two coplanar perpendicular surface monopoles i and j   c
c                                                                  c
c   called by: ortot                                               c
c                                                                  c
c   calls: gq1do, fumo, conso, gaus2, gaus4, gaus6, gaus24         c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   calculate the constants and initialize
c
      pi=atan(1.0)*4.0
      kd=2.0*pi*d
      kw1=2.0*pi*w1
      kw2=2.0*pi*w2
      kh1=2.0*pi*h1
      kh2=2.0*pi*h2
      khh=2.0*pi*hh
      a(1)=-i11+i21*cos(kh1)
      a(2)=-i21+i11*cos(kh1)
      gamma=-30.0/(2.0*kw1*kh2*sin(2.0*kw2)*sin(kh1))
c
      x1(1)=-(kw1+kw2)
      x2(1)=-abs(kw1-kw2)
      x1(2)=x2(1)
      x2(2)=-x2(1)
      x1(3)=x2(2)
      x2(3)=-x1(1)
c
      spt=-kd
      z12=0.0
      ng=24
      if (abs(d).gt.(2.0*(w1+w2)/2.0)) ng=6
      if (abs(d).gt.(4.0*(w1+w2)/2.0)) ng=4
      if (abs(d).gt.(6.0*(w1+w2)/2.0)) ng=2
c
c   calculate the mutual impedance
c
      do 10 i=1,2
      flag=i
      do 20 j=1,3
      if (x1(j).eq.x2(j)) goto 20
      call gq1do(kd,khh,kh1,kh2,z,typ,flag,x1(j),x2(j),
     +           kw1,kw2,i12,i22,spt,j,cj,ng)
      z12=z12+z*a(i)
      call conso(kd,khh,kh1,kh2,z,typ,flag,x1(j),x2(j),
     +           kw1,kw2,i12,i22,spt,j,cj)
      z12=z12+z*a(i)
   20 continue
   10 continue
      z12=z12*gamma
      return
      end
c
c   gq1do - calculates integral in one dimension
c
      subroutine gq1do(kd,khh,kh1,kh2,z,typ,flag,x1,x2,kw1,
     +                 kw2,i12,i22,spt,m,cj,ngaus)
      real    kd,x1,x2,kw1,kw2,xm,xr,p1,p2,x(24),w(24),dx,
     +        i12,i22,khh,kh1,kh2,pi,spt
      complex   z,r1,r2,cj
      integer typ,j,flag,ngaus,m
```

```
c
      if (ngaus.eq.24) call gaus24(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      if (ngaus.eq.4) call gaus4(x,w)
      if (ngaus.eq.2) call gaus2(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 j=1,ngaus
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      call funo(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,p1,r1,
     +          x1,x2,spt,cj,m)
      call funo(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,p2,r2,
     +          x1,x2,spt,cj,m)
      z=z+w(j)*(r1+r2)
  10  continue
      z=xr*z
      return
      end
c
c   funo calculates the integrand of one dim. integral
c
      subroutine  funo(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,x,z,
     +                 x1,x2,spt,cj,m)
      real   kd,khh,kh1,kh2,kw1,kw2,i12,i22,x,c1,c2,c3,c4,c6,
     +       g0,f03,f04,x1,x2,spt
      complex  z,p1,c5,c7,cj
      integer typ,flag,m
c
c   calculate the related parameters
c
      p1=(0.0,-1.0)
      c1=(flag)-2.0
      c2=khh+c1*kh1
      c3=sqrt((kd+x)**2+(c2+kh2)**2)
      c4=sqrt((kd+x)**2+c2**2)
      f03=sqrt((c2+kh2)**2)
      f04=sqrt(c2**2)
      c5=(cexp(p1*c3)-cexp(p1*c4))
      c7=(cexp(p1*f03)-cexp(p1*f04))
c
      if (m.eq.1) then
      c6=i12*(cos(kw2-kw1-x)-cos(2.0*kw2))
      c6=c6+i22*(1.0-cos(x+kw1+kw2))
      g0=i12*(cos(kw2+kd-kw1)-cos(2.0*kw2))
      g0=g0+i22*(1.0-cos(kw1+kw2-kd))
      endif
c
      if (m.eq.2) then
      if (kw1.gt.kw2) then
      c6=(1.0-cos(2.0*kw2))*(i12+i22)
      g0=c6
      else
      c6=i12*(cos(kw2-kw1-x)-cos(kw2+kw1-x))
      c6=c6-i22*(cos(kw2+kw1+x)-cos(kw2-kw1+x))
      g0=i12*(cos(kw2+kd-kw1)-cos(kw2+kw1+kd))
      g0=g0-i22*(cos(kw2+kw1-kd)-cos(kw2-kw1-kd))
      endif
      endif
c
      if (m.eq.3) then
      c6=i12*(1.0-cos(x-kw1-kw2))
      c6=c6+i22*(cos(x+kw2-kw1)-cos(2.0*kw2))
```

```
      g0=i12*(1.0-cos(-kw1-kw2-kd))
      g0=g0+i22*(cos(kw2-kw1-kd)-cos(2.0*kw2))
      endif
c     if ((kd+x).eq.(0.0)) then
      z=0.0
      else
      z=c5*c6
      z=(z-c7*g0)/(kd+x)
      endif
c     cj=c7*g0
c     return
      end
c
c   conso calculates the extracted singularity of integral
c
      subroutine conso(kd,khh,kh1,kh2,z,typ,flag,x1,x2,
     +           kw1,kw2,i12,i22,spt,j,cj)
      real  kd,khh,kh1,kh2,x1,x2,kw1,kw2,i12,i22,spt
      complex   z,cj
      integer typ,flag,j
c     if (spt.eq.x1) then
      z=log(kd+x2)
      else
      if (spt.eq.x2) then
      z=-log(abs(kd+x1))
      else
      z=log(abs((kd+x2)/(kd+x1)))
      endif
      endif
      z=z*cj
      return
      end
```

```
      subroutine orf(i11,i21,i12,i22,d,hh,t,w1,h1,w2,h2,z12)
      real  pi,i11,i21,i12,i22,d,hh,h1,w1,kd,kw1,kh1,khh,t,
     +         a(2),h2,w2,kw2,kh2,gamma,x1(3),x2(3),kt
      complex   z12,z
      integer       i,j,typ,flag
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine orf calculates the mutual impedance                      c
c   between two non-planar perpendicular surface monopoles i and j      c
c                                                                       c
c   called by: orftot                                                   c
c                                                                       c
c   calls: gq1dof, funof, conso, gaus6                                  c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c   calculate the constants and initialize
c
      pi=atan(1.0)*4.0
      kd=2.0*pi*d
      kw1=2.0*pi*w1
      kw2=2.0*pi*w2
      kh1=2.0*pi*h1
      kh2=2.0*pi*h2
      khh=2.0*pi*hh
      kt=2.0*pi*t
      a(1)=-i11+i21*cos(kh1)
      a(2)=-i21+i11*cos(kh1)
      gamma=-30.0/(2.0*kw1*kh2*sin(2.0*kw2)*sin(kh1))
c
      x1(1)=-(kw1+kw2)
      x2(1)=-abs(kw1-kw2)
      x1(2)=x2(1)
      x2(2)=-x2(1)
      x1(3)=x2(2)
      x2(3)=-x1(1)
      z12=0.0
c
c   calculate the mutual impedance
c
      do 10 i=1,2
      flag=i
      do 20 j=1,3
      if (x1(j).eq.x2(j)) goto 20
      call gq1dof(kt,kd,khh,kh1,kh2,z,typ,flag,x1(j),x2(j),
     +           kw1,kw2,i12,i22,j)
      z12=z12+z*a(i)
 20   continue
 10   continue
      z12=z12*gamma
      return
      end
c
c   gq1dof - calculates integral in one dimension
c
      subroutine gq1dof(kt,kd,khh,kh1,kh2,z,typ,flag,x1,x2,kw1,
     +                  kw2,i12,i22,m)
      real kd,x1,x2,kw1,kw2,xm,xr,pi,p2,x(24),w(24),dx,
     +        i12,i22,khh,kh1,kh2,pi,kt,dist
      complex z,r1,r2
      integer typ,j,flag,ngaus,m
c
      pi=atan(1.0)*4.0
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 j=1,ngaus
      dx=xr*x(j)
```

```
      p1=xm+dx
      p2=xm-dx
      call funof(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,p1,r1,
     +           x1,x2,m,kt)
      call funof(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,p2,r2,
     +           x1,x2,m,kt)
      z=z+w(j)*(r1+r2)
   10 continue
      z=xr*z
      return
      end
c
c   funof calculates the integrand of one dim. integral
c
      subroutine  funof(kd,khh,kh1,kh2,typ,flag,kw1,kw2,i12,i22,x,z,
     +                  x1,x2,m,kt)
      real kd,khh,kh1,kh2,kw1,kw2,i12,i22,x,c1,c2,c3,c4,c6,
     +        x1,x2,kt
      complex z,p1,c5
      integer typ,flag,m
c
c   calculate the related parameters
c
      p1=(0.0,-1.0)
      c1=flag-2.0
      c2=khh+c1*kh1
      c3=sqrt((kd+x)**2+(c2+kh2)**2+kt**2)
      c4=sqrt((kd+x)**2+c2**2+kt**2)
      c5=(cexp(p1*c3)-cexp(p1*c4))
c
      if (m.eq.1) then
      c6=i12*(cos(kw2-kw1-x)-cos(2.0*kw2))
      c6=c6+i22*(1.0-cos(x+kw1+kw2))
      endif
c
      if (m.eq.2) then
      if (kw1.gt.kw2) then
      c6=(1.0-cos(2.0*kw2))*(i12+i22)
      else
      c6=i12*(cos(kw2-kw1-x)-cos(kw2+kw1-x))
      c6=c6-i22*(cos(kw2+kw1+x)-cos(kw2-kw1+x))
      endif
      endif
c
      if (m.eq.3) then
      c6=i12*(1.0-cos(x-kw1-kw2))
      c6=c6+i22*(cos(x+kw2-kw1)-cos(2.0*kw2))
      endif
c
      z=c5*c6*(kd+x)/((kd+x)**2+kt**2)
c
      return
      end
```

```
      subroutine zcdxx(kd,khh,kl,kw,kh,i1,i2,z12)
      real    i1,i2,kd,khh,kl,kw,kh,x1,
     +        x2,a(2),gamma
      complex   z12,z,p1
      integer i,j,flag,ngaus
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zcdxx calculates the interaction between x-directed c
c   conductor and dielectric currents, which is Ex of the cond.    c
c   current                                                        c
c                                                                  c
c   called by: filcd                                               c
c                                                                  c
c   calls: gq1d20, fun20, gaus2, gaus6                             c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(kh*sin(2.0*kw))
      a(1)=i2*cos(2.0*kw)-i1
      a(2)=-i2+i1*cos(2.0*kw)
      x1=0.0
      x2=kh
      z12=0.0
      ngaus=6
      if (kl.gt.(3.*kh)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=2*i-3
      call gq1d20(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D20 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d20(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real    kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.2) call gaus2(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun20(kw,kd,khh,kl,flag,p1,z1)
      call fun20(kw,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun20(kw,kd,khh,kl,flag,x,z)
      real   kw,kd,khh,kl,x,c1,c2,c3,c4
      complex z,p1
      integer flag
c
      c1=kl-x
      c2=(flag)*kw
```

```
p1=(0.0,1.0)
c3=kd+c2
c4=sqrt(c1**2+c3**2+khh**2)
z=-cexp(-p1*c4)/c4
return
end
```

```
      subroutine zcdxy(kd,khh,kl,kw,kh,i1,i2,z12)
      real     i1,i2,kd,khh,kl,kw,kh,x1,gamma,
     +         x2,a(2)
      complex z12,z,p1
      integer i,j,flag,ngaus
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zcdxy calculates the interaction between x-directed c
c   conductor and y-directed dielectric currents,                  c
c   which is Ey of the conductor current.                          c
c   current                                                        c
c                                                                  c
c   called by: filcd, filvltn                                      c
c                                                                  c
c   calls: gq1d21, fun21, gaus2, gaus6                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(kh*sin(2.0*kw))
      a(1)=i2*cos(2.0*kw)-i1
      a(2)=-i2+i1*cos(2.0*kw)
      x1=0.0
      x2=kh
      z12=0.0
      ngaus=6
      if (kl.gt.(3.*kh)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=2*i-3
      call gq1d21(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D21 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d21(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real     kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex   z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.2) call gaus2(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun21(kw,kd,khh,kl,flag,p1,z1)
      call fun21(kw,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN21 CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun21(kw,kd,khh,kl,flag,x,z)
      real     kw,kd,khh,kl,x,c1,c2,c3,c4,c5
      complex   z,p1
      integer flag
c
```

```
c1=k1-x
c2=(flag)*kw
p1=(0.0,1.0)
c3=kd+c2
c4=sqrt(c1**2+c3**2+khh**2)
c5=khh**2+c1**2
z=khh*cexp(-p1*c4)*c3/(c4*c5)
return
end
```

```
      subroutine zcdxz(kd,khh,kl,kw,kh,i1,i2,z12)
      real    i1,i2,kd,khh,kl,kw,kh,x1,gamma,
     +        x2,a(2)
      complex   z12,z,p1
      integer i,j,ngaus,flag
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zcdxz calculates the interaction between x-directed c
c   conductor and z-directed dielectric currents,                  c
c   which is Ez of the conductor current.                          c
c   current                                                        c
c                                                                  c
c   called by: filcd, filvlt                                       c
c                                                                  c
c   calls: gq1d22, fun22, gaus2, gaus6                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(kh*sin(2.0*kw))
      a(1)=i2*cos(2.0*kw)-i1
      a(2)=-i2+i1*cos(2.0*kw)
      x1=0.0
      x2=kh
      z12=0.0
      ngaus=6
      if (kl.gt.(3.*kh)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=2*i-3
      call gq1d22(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D22 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d22(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real  kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.2) call gaus2(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun22(kw,kd,khh,kl,flag,p1,z1)
      call fun22(kw,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN22 CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun22(kw,kd,khh,kl,flag,x,z)
      real   kw,kd,khh,kl,x,c1,c2,c3,c4,c5
      complex z,p1
      integer flag
c
      c1=kl-x
```

```
c2=(flag)*kw
p1=(0.0,1.0)
c3=kd+c2
c4=sqrt(c1**2+c3**2+khh**2)
c5=khh**2+c1**2
z=c1*cexp(-p1*c4)*c3/(c4*c5)
return
end
```

```
      subroutine zcdzx(kd,khh,kl,kw,kh,i1,i2,z12)
      real  i1,i2,kd,khh,kl,kw,kh,x1,
     +        x2,a(2),gamma
      complex   z12,z,p1
      integer i,j,flag,ngaus
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zcdzx calculates the interaction between z-directed c
c   conductor and x-directed dielectric currents,                  c
c   which is Ex of the conductor current.                          c
c   current                                                        c
c                                                                  c
c   called by: filcd                                               c
c                                                                  c
c   calls: gq1d23, fun23, gaus2, gaus6                             c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(2.0*kw*sin(kh))
      a(1)=i2*cos(kh)-i1
      a(2)=-i2+i1*cos(kh)
      x1=-kw
      x2=-x1
      z12=0.0
      ngaus=6
      if (kd.gt.(3.*kw)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=i
      call gq1d23(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D23 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d23(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real  kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex   z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.6) call gaus6(x,w)
      if (ngaus.eq.2) call gaus2(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun23(kh,kd,khh,kl,flag,p1,z1)
      call fun23(kh,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN23 CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun23(kh,kd,khh,kl,flag,x,z)
      real   kh,kd,khh,kl,x,c1,c2,c3,c4,c5
      complex z,p1
      integer flag
c
      c1=kd-x
```

```
c2=(flag-2)*kh
p1=(0.0,1.0)
c3=k1+c2
c4=sqrt(c1**2+c3**2+khh**2)
c5=c1**2+khh**2
z=c1*cexp(-p1*c4)*c3/(c4*c5)
return
end
```

```
      subroutine zcdzy(kd,khh,kl,kw,kh,i1,i2,z12)
      real  i1,i2,kd,khh,kl,kw,kh,x1,
     +        x2,a(2),gamma
      complex z12,z,p1
      integer i,j,flag,ngaus
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    Subroutine zcdzy calculates the interaction between z-directed c
c    conductor and y-directed dielectric currents,                  c
c    which is Ey of the conductor current.                          c
c    current                                                        c
c                                                                   c
c    called by: filcd, filvltn                                      c
c    calls: gq1d25, fun25, gaus2, gaus6                             c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(2.0*kw*sin(kh))
      a(1)=i2*cos(kh)-i1
      a(2)=-i2+i1*cos(kh)
      x1=-kw
      x2=-x1
      z12=0.0
      ngaus=6
      if (kd.gt.(3.*kw)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=i
      call gq1d25(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D25 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d25(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.2) call gaus2(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun25(kh,kd,khh,kl,flag,p1,z1)
      call fun25(kh,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN25 CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun25(kh,kd,khh,kl,flag,x,z)
      real kh,kd,khh,kl,x,c1,c2,c3,c4
      complex z,p1
      integer flag
c
      c1=kd-x
```

```
c2=(flag-2)*kh
p1=(0.0,1.0)
c3=kl+c2
c4=sqrt(c1**2+c3**2+khh**2)
z=khh*cexp(-p1*c4)*c3/(c4*(c1**2+khh**2))
return
end
```

```
      subroutine zcdzz(kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,x1,
     +        x2,a(2),gamma
      complex z12,z,p1
      integer i,j,flag,ngaus
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zcdzz calculates the interaction between z-directed c
c   conductor and dielectric currents, which is Ez of the cond.    c
c   current                                                        c
c                                                                  c
c   called by: filcd, filvlt                                       c
c                                                                  c
c   calls: gq1d26, fun26, gaus2, gaus6                             c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0/(2.0*kw*sin(kh))
      a(1)=i2*cos(kh)-i1
      a(2)=-i2+i1*cos(kh)
      x1=-kw
      x2=-x1
      z12=0.0
      ngaus=6
      if (kd.gt.(3.*kw)) ngaus=2
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,2
      flag=i
      call gq1d26(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      z12=z12+z*a(i)
   10 continue
      z12=z12*gamma*p1
      return
      end
c
c     GQ1D26 - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1d26(kw,kh,kd,khh,kl,x1,x2,ngaus,flag,z)
      real kd,khh,kh,kl,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,kw
      complex z,z1,z2
      integer i,flag,ngaus
c
      if (ngaus.eq.2) call gaus2(x,w)
      if (ngaus.eq.6) call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun26(kh,kd,khh,kl,flag,p1,z1)
      call fun26(kh,kd,khh,kl,flag,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN26 CALCULATES THE INTEGRAND OF ONE DIMENSIONAL INTEGRAL
c
      subroutine  fun26(kh,kd,khh,kl,flag,x,z)
      real kh,kd,khh,kl,x,c1,c2,c3,c4
      complex z,p1
      integer flag
c
      c1=kd-x
      c2=(flag-2)*kh
```

```
p1=(0.0,1.0)
c3=k1+c2
c4=sqrt(c1**2+c3**2+khh**2)
z=-cexp(-p1*c4)/c4
return
end
```

```
      subroutine zdcxx(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    Subroutine zdcxx calculates the interaction between x-directed c
c    conductor and dielectric currents.                              c
c                                                                    c
c    called by: fildc                                                c
c                                                                    c
c    calls: zcdxx                                                    c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kl-0.5*kh).lt.(2.0*kh)) then
      m=4
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      endif
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
cc    CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdxx(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(kly*klz)
      return
      end
```

```
      subroutine zdcxy(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zdcxy calculates the interaction between x-directed c
c   conductor and y-directed dielectric currents.                   c
c                                                                   c
c   called by: fildc                                                c
c                                                                   c
c   calls: zcdxy                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kl-0.5*kh).lt.(2.0*kh)) then
      m=4
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      endif
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdxy(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(klx*klz)
      return
      end
```

```
      subroutine zdcxz(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zdcxz calculates the interaction between x-directed c
c   conductor and z-directed dielectric currents.                   c
c                                                                   c
c   called by: fildc                                                c
c                                                                   c
c   calls: zcdxz                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kl-0.5*kh).lt.(2.0*kh)) then
      m=4
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      endif
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdxz(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(klx*kly)
      return
      end
```

```
      subroutine zdczx(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zdczx calculates the interaction between z-directed c
c   conductor and x-directed dielectric currents.                  c
c                                                                  c
c   called by: fildc                                               c
c                                                                  c
c   calls: zcdzx                                                   c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kd).lt.(4.0*kw)) then
      m=4
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      endif
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdzx(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(kly*klz)
      return
      end
```

```
      subroutine zdczy(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    Subroutine zdczy calculates the interaction between z-directed c
c    conductor and y-directed dielectric currents.                   c
c                                                                    c
c    called by: fildc                                                c
c                                                                    c
c    calls: zcdzy                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kd).lt.(4.0*kw)) then
      m=4
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      endif
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdzy(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(klx*klz)
      return
      end
```

```
      subroutine zdczz(klx,kly,klz,kd,khh,kl,kw,kh,i1,i2,z12)
      real i1,i2,kd,khh,kl,kw,kh,klx,kly,klz,rx,rz
      complex z12,z,p1
      integer i,j,flag,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zdczz calculates the interaction between z-directed c
c   conductor and dielectric currents.                              c
c                                                                   c
c   called by: fildc                                                c
c                                                                   c
c   calls: zcdzz                                                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      z12=0.0
      if (abs(kd).lt.(4.0*kw)) then
      m=4
      if (abs(kl-0.5*kh).gt.(2.0*kh)) m=3
      if (abs(kl-0.5*kh).gt.(4.0*kh)) m=2
      if (abs(kl-0.5*kh).gt.(6.0*kh)) m=1
      endif
      if (abs(kd).gt.(4.*kw)) m=3
      if (abs(kd).gt.(8.*kw)) m=2
      if (abs(kd).gt.(12.*kw)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      do 10 i=1,nx
      do 10 j=1,nz
      rx=kd-klx/2.+(i-0.5)*klx/nx
      rz=kl-klz/2.+(j-0.5)*klz/nz
      call zcdzz(rx,khh,rz,kw,kh,i1,i2,z)
      z12=z12+z*klx*klz*kly/(nx*nz)
   10 continue
      z12=-z12/(klx*kly)
      return
      end
```

```
      subroutine zxx(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,
     +         gamma,x1,x2,y1,y2,z1,z2,kl,r,rc
      complex   z12,z,p1
      integer       i,typ,j,k,flag,app,nx,nz,napp,m,m1
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zxx calculates the interaction between x-directed   c
c   dielectric currents, which is Ex of x-directed dielectric      c
c   current                                                        c
c                                                                  c
c   called by: fildd                                               c
c                                                                  c
c   calls: zxxapp, gq3dxx, fun3xx1, gq2ddxx, fun2xx, fun2xx2,      c
c          fun2xx3, gaus6                                          c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      typ=0
      p1=(0.0,1.0)
      gamma=30.0
      z12=0.0
      app=1
      rc=sqrt(klx**2+klz**2+kly**2)
c
      if ((abs(kd).lt.klx).and.(abs(kl).lt.klz)) typ=1
c   if typ=1 use general form no approximation - used once
      x1=-klx/2.0
      x2=-x1
      y1=-kly/2.0
      y2=-y1
      z1=-klz/2.0
      z2=-z1
      r=sqrt(kd**2+khh**2+kl**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if ((app.eq.1).and.(typ.eq.0)) then
c   use approximate interaction
      call zxxapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      else
c   use general form
      if (typ.eq.0) then
      call gq3ddxx(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=4
      call gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,y1,y2,z1,z2)
      z12=z12+z
      else
      call gq3ddxx(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=1
      call gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,y1,y2,z1,z2)
      z12=z12+z
      flag=2
```

```
      call gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,z1,z2)
      z12=z12+z
      flag=3
      call gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      z12=z12+z
      flag=4
      call gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,y1,y2,z1,z2)
      z12=z12+z
      endif
      endif
c
      z12=z12*gamma
      return
      end
c
      subroutine zxxapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cx
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cx=d/r
      z12=z12+(1-cx**2+p1/r*(1-p1/r)*(3*cx**2-1))*cexp(-p1*r)/r
  10  continue
      z12=-p1*z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ3DDXX - CALCULATES INTEGRAL IN THREE DIMENSIONS
c
      subroutine gq3ddxx(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,s1,s2)
      real kd,khh,kl,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,s1,s2,ds,sm,sr,r1,r2
      complex z,z1,z2,z3,z4,z5,z6,z7,z8
      integer j,i,k,typ,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      sm=0.50*(s1+s2)
      sr=0.50*(s2-s1)
      z=0.0
      do 10 i=1,6
      ds=sr*x(i)
      r1=sm+ds
      r2=sm-ds
      do 20 j=1,6
      dy=yr*x(j)
      p1=ym+dy
      p2=ym-dy
      do 30 k=1,6
      dx=xr*x(k)
      q1=xm+dx
      q2=xm-dx
      call fun3xx1(kd,khh,kl,r1,p1,q1,z1,typ)
      call fun3xx1(kd,khh,kl,r1,p1,q2,z2,typ)
      call fun3xx1(kd,khh,kl,r1,p2,q1,z3,typ)
      call fun3xx1(kd,khh,kl,r1,p2,q2,z4,typ)
```

```
      call fun3xx1(kd,khh,kl,r2,p1,q1,z5,typ)
      call fun3xx1(kd,khh,kl,r2,p1,q2,z6,typ)
      call fun3xx1(kd,khh,kl,r2,p2,q1,z7,typ)
      call fun3xx1(kd,khh,kl,r2,p2,q2,z8,typ)
      z=z+w(i)*w(j)*w(k)*(z1+z2+z3+z4+z5+z6+z7+z8)
   30 continue
   20 continue
   10 continue
      z=xr*yr*sr*z
      return
      end
c
cc    FUN3XX1 CALCULATES INTEGRAND OF THREE DIM. INTEGRAL
c
      subroutine  fun3xx1(kd,khh,kl,z,y,x,zz,typ)
      real kd,khh,kl,c1,x,y,z
      complex zz,p1
      integer typ
c
cc    CALCULATE THE RELATED PARAMETERS
c
      p1=(0.0,-1.0)
      zz=0.0
      c1=sqrt((kd-x)**2+(khh-y)**2+(kl-z)**2)
c
cc    CALCULATE THE INTEGRAND
c
      if (typ.eq.0) then
      zz=p1*cexp(p1*c1)/c1
      else
      zz=cexp(p1*c1)/2.0
      endif
      return
      end
c
cc    GQ2DDXI - CALCULATES INTEGRAL IN TWO DIMENSIONS
c
      subroutine gq2ddxx(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      real kd,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,klx,kly,klz,khh,kl
      complex z,z1,z2,z3,z4
      integer typ,j,i,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      z=0.0
      do 10 i=1,6
      dy=yr*x(i)
      q1=ym+dy
      q2=ym-dy
      do 20 j=1,6
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      call fun2xx(khh,kd,kl,typ,flag,klx,kly,klz,q1,p1,z1)
      call fun2xx(khh,kd,kl,typ,flag,klx,kly,klz,q1,p2,z2)
      call fun2xx(khh,kd,kl,typ,flag,klx,kly,klz,q2,p1,z3)
      call fun2xx(khh,kd,kl,typ,flag,klx,kly,klz,q2,p2,z4)
      z=z+w(i)*w(j)*(z1+z2+z3+z4)
   20 continue
   10 continue
      z=xr*yr*z
      return
      end
c
cc    FUN2XX CALCULATES THE INTEGRAND OF DOUBLE INTEGRAL
```

```
c
      subroutine fun2xx(khh,kd,kl,typ,flag,klx,kly,klz,q,p,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,c1,c2,q,p
      complex zz,p1,zp
      integer typ,flag,i
c
      p1=(0.0,1.0)
      zz=0.0
      do 10 i=-1,1,2
c
      if (flag.eq.1) then
      x=kd-(i)*klx/2.0
      y=khh-p
      z=kl-q
      call fun2xx2(x,y,z,zp)
      zz=zz+zp*x*(i)
      endif
c
      if (flag.eq.2) then
      x=kd-p
      y=khh-(i)*kly/2.0
      z=kl-q
      call fun2xx2(x,y,z,zp)
      zz=zz+zp*y*(i)
      endif
c
      if (flag.eq.3) then
      x=kd-p
      y=khh-q
      z=kl-(i)*klz/2.0
      call fun2xx2(x,y,z,zp)
      zz=zz+zp*z*(i)
      endif
c
      if (flag.eq.4) then
      x=kd-(i)*klx/2.0
      y=khh-p
      z=kl-q
      call fun2xx3(x,y,z,zp)
      zz=zz+zp*x*(i)
      endif
c
 10   continue
      return
      end
c
      subroutine fun2xx2(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=p1*cexp(-p1*c1)/(2.0*c1)
      return
      end
c
      subroutine fun2xx3(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)*((1.0/c1**2)-(p1/c1**3))
      return
      end
```

```
      subroutine zxy(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,kl,r,rc,
     +         gamma,x1,x2,y1,y2,z1,z2
      complex   z12,z,p1
      integer       i,typ,j,k,app,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zxy calculates the interaction between x-directed    c
c   and y-directed dielectric currents, which is Ey of              c
c   x-directed dielectric current                                   c
c                                                                   c
c   called by: fildd                                                c
c                                                                   c
c   calls: zxyapp, gq1dxy, fun1xy, fun2xy, gaus6                    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0
      z12=0.0
c   approximate form is used to save time
      app=1
c
      r=sqrt(kd**2+khh**2+kl**2)
      rc=sqrt(klx**2+kly**2+klz**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if (app.eq.1) then
c   use approximate interaction
      call zxyapp(kd,khh,kl,klx,kly,klz,nx,nz,z)
      else
c   use general form
      z1=-klz/2.0
      z2=-z1
      call gq1dxy(khh,kl,kd,z,z1,z2,klx,kly,klz)
      endif
      z12=z*gamma*p1
      return
      end
c
      subroutine zxyapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cx,cy
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cx=d/r
      cy=khh/r
```

```
      z12=z12+cx*cy*(-1+(3*p1/r)*(1-p1/r))*cexp(-p1*r)/r
   10 continue
      z12=z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ1DXY - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1dxy(khh,kl,kd,z,x1,x2,klx,kly,klz)
      real kd,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,
     +        klx,kly,klz,khh,kl
      complex z,z1,z2
      integer j,i,ngaus
c
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun1xy(khh,kd,kl,klx,kly,klz,p1,z1)
      call fun1xy(khh,kd,kl,klx,kly,klz,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN1XY CALCULATES INTEGRAND OF ONE DIM. INTEGRAL
c
      subroutine fun1xy(khh,kd,kl,klx,kly,klz,z,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,zl
      complex zz,p1,zp
      integer i,j
c
      p1=(0.0,1.0)
      zz=0.0
      zl=kl-z
      do 10 i=1,2
      do 10 j=1,2
      x=kd-(2*i-3)*klx/2.0
      y=khh-(2*j-3)*kly/2.0
      call fun2xy(x,y,zl,zp)
      zz=zz+zp*(2*i-3)*(2*j-3)
   10 continue
      return
      end
c
      subroutine fun2xy(x,y,z,zp)
      real x,y,z,c1
      complex p1,zp
c
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)/c1
      return
      end
```

```
      subroutine zxz(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,r,rc,
     +         gamma,x1,x2,y1,y2,z1,z2,kl
      complex   z12,z,p1
      integer       i,j,k,app,nx,nz,napp,m
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zxz calculates the interaction between x-directed    c
c   and z-directed dielectric currents, which is Ez of              c
c   x-directed dielectric current                                   c
c                                                                   c
c   called by: fildd                                                c
c                                                                   c
c   calls: zxzapp, gq1dxz, fun1xz, fun2xz, gaus6                    c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0
      z12=0.0
c   approximate form is used to save time
      app=1
c
      r=sqrt(kd**2+khh**2+kl**2)
      rc=sqrt(klx**2+kly**2+klz**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if (app.eq.1) then
c   use approximate interaction
      call zxzapp(kd,khh,kl,klx,kly,klz,nx,nz,z)
      else
c   use general form
      y1=-kly/2.0
      y2=-y1
      call gq1dxz(khh,kl,kd,z,y1,y2,klx,kly,klz)
      endif
      z12=z*gamma*p1
      return
      end
c
      subroutine zxzapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cx,cz
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cx=d/r
      cz=l/r
```

```
      z12=z12+cx*cz*(-1+3*p1/r*(1-p1/r))*cexp(-p1*r)/r
   10 continue
      z12=z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ1DXZ - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1dxz(khh,kl,kd,z,x1,x2,klx,kly,klz)
      real kd,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,
     +       klx,kly,klz,khh,kl
      complex z,z1,z2
      integer j,i,ngaus
c
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun1xz(khh,kd,kl,klx,kly,klz,p1,z1)
      call fun1xz(khh,kd,kl,klx,kly,klz,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN1XZ CALCULATES INTEGRAND OF ONE DIM. INTEGRAL
c
      subroutine fun1xz(khh,kd,kl,klx,kly,klz,y,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,yl
      complex zz,p1,zp
      integer i,j
c
      p1=(0.0,1.0)
      zz=0.0
      yl=khh-y
      do 10 i=1,2
      do 10 j=1,2
      x=kd-(2*i-3)*klx/2.0
      z=kl-(2*j-3)*klz/2.0
      call fun2xz(x,yl,z,zp)
      zz=zz+zp*(2*i-3)*(2*j-3)
   10 continue
      return
      end
c
      subroutine fun2xz(x,y,z,zp)
      real x,y,z,c1
      complex p1,zp
c
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)/c1
      return
      end
```

```
      subroutine zyy(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,r,rc,
     +         gamma,x1,x2,y1,y2,z1,z2,kl
      complex   z12,z,p1
      integer       i,typ,j,k,flag,app,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zyy calculates the interaction between y-directed   c
c   dielectric currents, which is Ey of y-directed dielectric      c
c   current                                                        c
c                                                                  c
c   called by: fildd                                               c
c                                                                  c
c   calls: zyyapp, gq3dyy, fun3yy1, gq2ddyy, fun2yy, fun2yy2,      c
c          fun2yy3, gaus6                                          c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      typ=0
      p1=(0.0,1.0)
      gamma=30.0
      z12=0.0
      app=1
c
      if ((abs(kd).lt.klx).and.(abs(kl).lt.klz)) typ=1
c   if typ=1 use general form no approximation - used once
      r=sqrt(kd**2+khh**2+kl**2)
      rc=sqrt(klx**2+kly**2+klz**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
      x1=-klx/2.0
      x2=-x1
      y1=-kly/2.0
      y2=-y1
      z1=-klz/2.0
      z2=-z1
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if ((app.eq.1).and.(typ.eq.0)) then
c   use approximate interaction
      call zyyapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      else
c   use general form
      if (typ.eq.0) then
      call gq3ddyy(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=4
      call gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,z1,z2)
      z12=z12+z
      else
      call gq3ddyy(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=1
      call gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,y1,y2,z1,z2)
      z12=z12+z
```

```
      flag=2
      call gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,z1,z2)
      z12=z12+z
      flag=3
      call gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      z12=z12+z
      flag=4
      call gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,z1,z2)
      z12=z12+z
      endif
      endif
c
      z12=z12*gamma
      return
      end
c
      subroutine zyyapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cy
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cy=khh/r
      z12=z12+(1-cy**2+p1/r*(1-p1/r)*(3*cy**2-1))*cexp(-p1*r)/r
   10 continue
      z12=-p1*z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ3DDYY - CALCULATES INTEGRAL IN THREE DIMENSIONS
c
      subroutine gq3ddyy(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,s1,s2)
      real kd,khh,kl,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,s1,s2,ds,sm,sr,r1,r2
      complex z,z1,z2,z3,z4,z5,z6,z7,z8
      integer j,i,k,typ,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      sm=0.50*(s1+s2)
      sr=0.50*(s2-s1)
      z=0.0
      do 10 i=1,6
      ds=sr*x(i)
      r1=sm+ds
      r2=sm-ds
      do 20 j=1,6
      dy=yr*x(j)
      p1=ym+dy
      p2=ym-dy
      do 30 k=1,6
      dx=xr*x(k)
      q1=xm+dx
      q2=xm-dx
      call fun3yy1(kd,khh,kl,r1,p1,q1,z1,typ)
      call fun3yy1(kd,khh,kl,r1,p1,q2,z2,typ)
      call fun3yy1(kd,khh,kl,r1,p2,q1,z3,typ)
```

```
      call fun3yy1(kd,khh,kl,r1,p2,q2,z4,typ)
      call fun3yy1(kd,khh,kl,r2,p1,q1,z5,typ)
      call fun3yy1(kd,khh,kl,r2,p1,q2,z6,typ)
      call fun3yy1(kd,khh,kl,r2,p2,q1,z7,typ)
      call fun3yy1(kd,khh,kl,r2,p2,q2,z8,typ)
      z=z+w(i)*w(j)*w(k)*(z1+z2+z3+z4+z5+z6+z7+z8)
   30 continue
   20 continue
   10 continue
      z=xr*yr*sr*z
      return
      end
c
c     FUN3YY1 CALCULATES INTEGRAND OF THREE DIM. INTEGRAL
c
      subroutine  fun3yy1(kd,khh,kl,z,y,x,zz,typ)
      real kd,khh,kl,c1,x,y,z
      complex zz,p1
      integer typ
c
c     CALCULATE THE RELATED PARAMETERS
c
      p1=(0.0,-1.0)
      zz=0.0
      c1=sqrt((kd-x)**2+(khh-y)**2+(kl-z)**2)
c
c     CALCULATE THE INTEGRAND
c
      if (typ.eq.0) then
      zz=p1*cexp(p1*c1)/c1
      else
      zz=cexp(p1*c1)/2.0
      endif
      return
      end
c
c     GQ2DDYY - CALCULATES INTEGRAL IN TWO DIMENSIONS
c
      subroutine gq2ddyy(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      real kd,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,klx,kly,klz,khh,kl
      complex z,z1,z2,z3,z4
      integer typ,j,i,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      z=0.0
      do 10 i=1,6
      dy=yr*x(i)
      q1=ym+dy
      q2=ym-dy
      do 20 j=1,6
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      call fun2yy(khh,kd,kl,typ,flag,klx,kly,klz,q1,p1,z1)
      call fun2yy(khh,kd,kl,typ,flag,klx,kly,klz,q1,p2,z2)
      call fun2yy(khh,kd,kl,typ,flag,klx,kly,klz,q2,p1,z3)
      call fun2yy(khh,kd,kl,typ,flag,klx,kly,klz,q2,p2,z4)
      z=z+w(i)*w(j)*(z1+z2+z3+z4)
   20 continue
   10 continue
      z=xr*yr*z
      return
      end
```

```
c
c     FUN2YY CALCULATES INTEGRAND OF DOUBLE INTEGRALS
c
      subroutine fun2yy(khh,kd,kl,typ,flag,klx,kly,klz,q,p,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,c1,c2,q,p
      complex zz,p1,zp
      integer typ,flag,i
c
      p1=(0.0,1.0)
      zz=0.0
      do 10 i=-1,1,2
c
      if (flag.eq.1) then
      x=kd-(i)*klx/2.0
      y=khh-p
      z=kl-q
      call fun2yy2(x,y,z,zp)
      zz=zz+zp*x*(i)
      endif
c
      if (flag.eq.2) then
      x=kd-p
      y=khh-(i)*kly/2.0
      z=kl-q
      call fun2yy2(x,y,z,zp)
      zz=zz+zp*y*(i)
      endif
c
      if (flag.eq.3) then
      x=kd-p
      y=khh-q
      z=kl-(i)*klz/2.0
      call fun2yy2(x,y,z,zp)
      zz=zz+zp*z*(i)
      endif
c
      if (flag.eq.4) then
      x=kd-p
      y=khh-(i)*kly/2.0
      z=kl-q
      call fun2yy3(x,y,z,zp)
      zz=zz+zp*y*(i)
      endif
c
 10   continue
      return
      end
c
      subroutine fun2yy2(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=p1*cexp(-p1*c1)/(2.0*c1)
      return
      end
c
      subroutine fun2yy3(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)*((1.0/c1**2)-(p1/c1**3))
      return
      end
```

```
      subroutine zyz(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,r,rc,
     +         gamma,x1,x2,y1,y2,z1,z2,kl
      complex   z12,z,p1
      integer       i,j,k,app,nx,nz,napp,m
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zyz calculates the interaction between y-directed    c
c   and z-directed dielectric currents, which is Ez of              c
c   y-directed dielectric current                                   c
c                                                                   c
c   called by: fildd                                                c
c                                                                   c
c   calls: zyzapp, gq1dyz, fun1yz, fun2yz, gaus6                    c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      p1=(0.0,1.0)
      gamma=-30.0
      z12=0.0
c   approximate form is used to save time
      app=1
c
      r=sqrt(kd**2+khh**2+kl**2)
      rc=sqrt(klx**2+kly**2+klz**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if (app.eq.1) then
c   use approximate interaction
      call zyzapp(kd,khh,kl,klx,kly,klz,nx,nz,z)
      else
c   use general form
      x1=-klx/2.0
      x2=-x1
      call gq1dyz(khh,kl,kd,z,x1,x2,klx,kly,klz)
      z12=z*gamma*p1
      endif
      return
      end
c
      subroutine zyzapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cy,cz
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cy=khh/r
      cz=kl/r
```

```
      z12=z12+cy*cz*(-1+3*p1/r*(1-p1/r))*cexp(-p1*r)/r
   10 continue
      z12=z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ1DYZ - CALCULATES INTEGRAL IN ONE DIMENSION
c
      subroutine gq1dyz(khh,kl,kd,z,x1,x2,klx,kly,klz)
      real kd,x1,x2,xm,xr,p1,p2,x(24),w(24),dx,
     +        klx,kly,klz,khh,kl
      complex z,z1,z2
      integer j,i,ngaus
c
      ngaus=6
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      z=0.0
      do 10 i=1,ngaus
      dx=xr*x(i)
      p1=xm+dx
      p2=xm-dx
      call fun1yz(khh,kd,kl,klx,kly,klz,p1,z1)
      call fun1yz(khh,kd,kl,klx,kly,klz,p2,z2)
      z=z+w(i)*(z1+z2)
   10 continue
      z=xr*z
      return
      end
c
c     FUN1YZ CALCULATES INTEGRAND OF ONE DIM. INTEGRAL
c
      subroutine fun1yz(khh,kd,kl,klx,kly,klz,x,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,xl
      complex zz,p1,zp
      integer i,j
c
      p1=(0.0,1.0)
      zz=0.0
      xl=kd-x
      do 10 i=1,2
      do 10 j=1,2
      y=khh-(2*i-3)*kly/2.0
      z=kl-(2*j-3)*klz/2.0
      call fun2yz(xl,y,z,zp)
      zz=zz+zp*(2*i-3)*(2*j-3)
   10 continue
      return
      end
c
      subroutine fun2yz(x,y,z,zp)
      real x,y,z,c1
      complex p1,zp
c
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)/c1
      return
      end
```

```
      subroutine zzz(klx,kly,klz,kd,khh,kl,z12)
      real  klx,kly,klz,kd,khh,r,rc,
     +         gamma,x1,x2,y1,y2,z1,z2,kl
      complex   z12,z,p1
      integer       i,typ,j,k,flag,app,nx,nz,napp,m
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine zzz calculates the interaction between z-directed    c
c   dielectric currents, which is Ez of z-directed dielectric       c
c   current                                                         c
c                                                                   c
c   called by: fildd                                                c
c                                                                   c
c   calls: zzzapp, gq3dzz, fun3zz1, gq2ddzz, fun2zz, fun2zz2,       c
c          fun2zz3, gaus6                                           c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c     CALCULATE THE CONSTANTS AND INITIALIZE
c
      typ=0
      p1=(0.0,1.0)
      gamma=30.0
      z12=0.0
      app=1
c
      if ((abs(kd).lt.klx).and.(abs(kl).lt.klz)) typ=1
c   if typ=1 use general form no approximation - used once
      x1=-klx/2.0
      x2=-x1
      y1=-kly/2.0
      y2=-y1
      z1=-klz/2.0
      z2=-z1
      r=sqrt(kd**2+khh**2+kl**2)
      rc=sqrt(klx**2+kly**2+klz**2)
      m=4
      if (r.gt.(3.*rc)) m=3
      if (r.gt.(6.*rc)) m=2
      if (r.gt.(9.*rc)) m=1
      if (klx.ge.klz) then
      napp=int(klx/klz+0.5)
      nz=m
      nx=m*napp
      else
      napp=int(klz/klx+0.5)
      nx=m
      nz=m*napp
      endif
c
c     CALCULATE THE MUTUAL IMPEDANCE
c
      if ((app.eq.1).and.(typ.eq.0)) then
c   use approximate interaction
      call zzzapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      else
c   use general form
      if (typ.eq.0) then
      call gq3ddzz(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=4
      call gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      z12=z12+z
      else
      call gq3ddzz(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,z1,z2)
      z12=z12+z
      flag=1
      call gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,y1,y2,z1,z2)
      z12=z12+z
      flag=2
```

```
      call gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,z1,z2)
      z12=z12+z
      flag=3
      call gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      z12=z12+z
      flag=4
      call gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      z12=z12+z
      endif
      endif
c
      z12=z12*gamma
      return
      end
c
      subroutine zzzapp(kd,khh,kl,klx,kly,klz,nx,nz,z12)
      real kd,khh,kl,klx,kly,klz,d,l,r,cz
      complex z12,p1
      integer nx,nz,i,j
c   calculates approximate interaction
      z12=0.0
      p1=(0.0,1.0)
      do 10 i=1,nx
      do 10 j=1,nz
      d=kd+klx/2.0+(0.5-i)*klx/nx
      l=kl+klz/2.0+(0.5-j)*klz/nz
      r=sqrt(d**2+khh**2+l**2)
      cz=l/r
      z12=z12+(1-cz**2+p1/r*(1-p1/r)*(3*cz**2-1))*cexp(-p1*r)/r
   10 continue
      z12=-p1*z12*klx*klz*kly/(nx*nz)
      return
      end
c
c     GQ3DDZZ - CALCULATES INTEGRAL IN THREE DIMENSIONS
c
      subroutine gq3ddzz(khh,kl,kd,z,typ,flag,x1,x2,y1,y2,s1,s2)
      real kd,khh,kl,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,s1,s2,ds,sm,sr,r1,r2
      complex z,z1,z2,z3,z4,z5,z6,z7,z8
      integer j,i,k,typ,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      sm=0.50*(s1+s2)
      sr=0.50*(s2-s1)
      z=0.0
      do 10 i=1,6
      ds=sr*x(i)
      r1=sm+ds
      r2=sm-ds
      do 20 j=1,6
      dy=yr*x(j)
      p1=ym+dy
      p2=ym-dy
      do 30 k=1,6
      dx=xr*x(k)
      q1=xm+dx
      q2=xm-dx
      call fun3zz1(kd,khh,kl,r1,p1,q1,z1,typ)
      call fun3zz1(kd,khh,kl,r1,p1,q2,z2,typ)
      call fun3zz1(kd,khh,kl,r1,p2,q1,z3,typ)
      call fun3zz1(kd,khh,kl,r1,p2,q2,z4,typ)
```

```
      call fun3zz1(kd,khh,kl,r2,p1,q1,z5,typ)
      call fun3zz1(kd,khh,kl,r2,p1,q2,z6,typ)
      call fun3zz1(kd,khh,kl,r2,p2,q1,z7,typ)
      call fun3zz1(kd,khh,kl,r2,p2,q2,z8,typ)
      z=z+w(i)*w(j)*w(k)*(z1+z2+z3+z4+z5+z6+z7+z8)
   30 continue
   20 continue
   10 continue
      z=xr*yr*sr*z
      return
      end
c
cc    FUN3ZZ1 CALCULATES INTEGRAND OF THREE DIM. INTEGRAL
c
      subroutine  fun3zz1(kd,khh,kl,z,y,x,zz,typ)
      real kd,khh,kl,c1,x,y,z
      complex zz,p1
      integer typ
c
cc    CALCULATE THE RELATED PARAMETERS
c
      p1=(0.0,-1.0)
      zz=0.0
      c1=sqrt((kd-x)**2+(khh-y)**2+(kl-z)**2)
c
cc    CALCULATE THE INTEGRAND
c
      if (typ.eq.0) then
      zz=p1*cexp(p1*c1)/c1
      else
      zz=cexp(p1*c1)/2.0
      endif
      return
      end
c
cc    GQ2DDZZ - CALCULATES INTEGRAL IN TWO DIMENSIONS
c
      subroutine gq2ddzz(klx,kly,klz,khh,kl,kd,z,typ,flag,x1,x2,y1,y2)
      real kd,x1,x2,xm,xr,p1,p2,q1,q2,x(24),w(24),dx,dy,
     +        y1,y2,ym,yr,klx,kly,klz,khh,kl
      complex z,z1,z2,z3,z4
      integer typ,j,i,flag
c
      call gaus6(x,w)
      xm=0.50*(x1+x2)
      xr=0.50*(x2-x1)
      ym=0.50*(y1+y2)
      yr=0.50*(y2-y1)
      z=0.0
      do 10 i=1,6
      dy=yr*x(i)
      q1=ym+dy
      q2=ym-dy
      do 20 j=1,6
      dx=xr*x(j)
      p1=xm+dx
      p2=xm-dx
      call fun2zz(khh,kd,kl,typ,flag,klx,kly,klz,q1,p1,z1)
      call fun2zz(khh,kd,kl,typ,flag,klx,kly,klz,q1,p2,z2)
      call fun2zz(khh,kd,kl,typ,flag,klx,kly,klz,q2,p1,z3)
      call fun2zz(khh,kd,kl,typ,flag,klx,kly,klz,q2,p2,z4)
      z=z+w(i)*w(j)*(z1+z2+z3+z4)
   20 continue
   10 continue
      z=xr*yr*z
      return
      end
c
cc    FUN2ZZ CALCULATES THE INTEGRAND OF DOUBLE INTEGRAL
```

```
c
      subroutine fun2zz(khh,kd,kl,typ,flag,klx,kly,klz,q,p,zz)
      real kd,kl,khh,klx,kly,klz,x,y,z,c1,c2,q,p
      complex zz,p1,zp
      integer typ,flag,i
c
      p1=(0.0,1.0)
      zz=0.0
      do 10 i=-1,1,2
c
      if (flag.eq.1) then
      x=kd-(i)*klx/2.0
      y=khh-p
      z=kl-q
      call fun2zz2(x,y,z,zp)
      zz=zz+zp*x*(i)
      endif
c
      if (flag.eq.2) then
      x=kd-p
      y=khh-(i)*kly/2.0
      z=kl-q
      call fun2zz2(x,y,z,zp)
      zz=zz+zp*y*(i)
      endif
c
      if (flag.eq.3) then
      x=kd-p
      y=khh-q
      z=kl-(i)*klz/2.0
      call fun2zz2(x,y,z,zp)
      zz=zz+zp*z*(i)
      endif
c
      if (flag.eq.4) then
      x=kd-p
      y=khh-q
      z=kl-(i)*klz/2.0
      call fun2zz3(x,y,z,zp)
      zz=zz+zp*z*(i)
      endif
c
  10  continue
      return
      end
c
      subroutine fun2zz2(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=p1*cexp(-p1*c1)/(2.0*c1)
      return
      end
c
      subroutine fun2zz3(x,y,z,zp)
      real x,y,z,c1
      complex zp,p1
      p1=(0.0,1.0)
      c1=sqrt(x**2+y**2+z**2)
      zp=cexp(-p1*c1)*((1.0/c1**2)-(p1/c1**3))
      return
      end
```

```
      subroutine symxx(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2,js,is
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symxx investigates symmetry for zxx                     c
c   If there is even symmetry, ok=1, mult=1                            c
c   If there is  odd symmetry, ok=1, mult=-1                           c
c                                                                      c
c   called by: zxx                                                     c
c                                                                      c
c   calls: none                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      m1=nl*nw
      m2=m1*nh
      mult=1.0
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.gt.jl) then
      js=j-1
      is=i-1
      ok=1
      endif
      endif
      endif
      endif
      else
c
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
c
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      js=j-m1
      is=i-m1
      ok=1
```

```
      endif
      else
      endif
c
      endif
      endif
      return
      end
```

```
      subroutine symxy(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2,js,is
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symxy investigates symmetry for zxy                   c
c   If there is even symmetry, ok=1, mult=1                          c
c   If there is  odd symmetry, ok=1, mult=-1                         c
c                                                                    c
c   called by: zxy                                                   c
c                                                                    c
c   calls: none                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      m1=nl*nw
      m2=m1*nh
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      endif
      endif
      endif
      endif
      else
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      ok=1
      endif
      endif
```

```
else
js=j-m1
is=i-m1
ok=1
endif
else
if ((jh.ne.ih).and.(jh.gt.ih)) then
js=j-(jh-ih)*m1
is=i+(jh-ih)*m1
mult=-1.0
ok=1
endif
endif
endif
endif
return
end
```

```
      subroutine symxz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer js,is,j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symxz investigates symmetry for zxz                     c
c   If there is even symmetry, ok=1, mult=1                            c
c   If there is  odd symmetry, ok=1, mult=-1                           c
c                                                                      c
c   called by: zxz                                                     c
c                                                                      c
c   calls: none                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      ok=0
      m1=nl*nw
      m2=m1*nh
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.gt.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      ok=1
      endif
      endif
      endif
      endif
      else
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.1) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      mult=-1.0
      ok=1
```

```
endif
endif
else
js=j-m1
is=i-m1
ok=1
endif
else
if ((jh.ne.ih).and.(jh.gt.ih)) then
js=j-(jh-ih)*m1
is=i+(jh-ih)*m1
ok=1
endif
endif
endif
endif
return
end
```

```
      subroutine symyy(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2,js,is
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symyy investigates symmetry for zyy                     c
c   If there is even symmetry, ok=1, mult=1                            c
c   If there is  odd symmetry, ok=1, mult=-1                           c
c                                                                      c
c   called by: zyy                                                     c
c                                                                      c
c   calls: none                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      m1=nl*nw
      m2=m1*nh
      mult=1.0
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.(m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.gt.jl) then
      js=j-1
      is=i-1
      ok=1
      endif
      endif
      endif
      endif
      else
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.(m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      js=j-m1
      is=i-m1
      ok=1
      endif
      else
```

```
endif
endif
endif
return
end
```

```
      subroutine symyz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer js,is,j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symyz investigates symmetry for zyz                     c
c   If there is even symmetry, ok=1, mult=1                            c
c   If there is  odd symmetry, ok=1, mult=-1                           c
c                                                                      c
c   called by: zyz                                                     c
c                                                                      c
c   calls: none                                                        c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      m1=nl*nw
      m2=m1*nh
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.(m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.gt.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      endif
      endif
      else
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.(m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      js=j-m1
```

```
is=i-m1
ok=1
endif
else
if ((jh.ne.ih).and.(jh.gt.ih)) then
js=j-(jh-ih)*m1
is=i+(jh-ih)*m1
mult=-1.0
ok=1
endif
endif
endif
endif
return
end
```

```
      subroutine symzz(j,i,jt,jh,jw,jl,it,ih,iw,il,nl,nw,nh,
     +                 js,is,ok,mult)
      real    mult
      integer j,i,jt,jh,jw,jl,it,ih,iw,il,ok,m1,m2,js,is
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine symzz investigates symmetry for zzz                    c
c   If there is even symmetry, ok=1, mult=1                           c
c   If there is  odd symmetry, ok=1, mult=-1                          c
c                                                                     c
c   called by: zzz                                                    c
c                                                                     c
c   calls: none                                                       c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      m1=nl*nw
      m2=m1*nh
      mult=1.0
      if ((jh.eq.ih).and.(jh.eq.0))  then
      if (jw.eq.0) then
      if (j.gt.(2*m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      if (iw.gt.0) then
      js=j-nl
      is=i-nl
      ok=1
      else
      if (jl.gt.1) then
      if (il.gt.jl) then
      js=j-1
      is=i-1
      ok=1
      endif
      endif
      endif
      endif
      else
      if ((jh.eq.ih).and.(jh.gt.0)) then
      js=j-m1
      is=i-m1
      ok=1
      else
      if ((jh.ne.ih).and.(jh.lt.ih)) then
      if (jh.eq.0) then
      if (j.gt.(2*m2+1)) then
      if (il.ge.jl) then
      js=j-1
      is=i-1
      ok=1
      else
      js=j-(jl-il)
      is=i+(jl-il)
      ok=1
      endif
      endif
      else
      js=j-m1
      is=i-m1
      ok=1
      endif
      else
```

```
endif
endif
endif
return
end
```

```
      subroutine gaus2(z,w)
      real z(24),w(24)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine gaus2 contains the roots of 4th order                c
c   Legendre polynomial                                             c
c                                                                   c
c   called by: various integration routines prefixed gq             c
c                                                                   c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      z(1) =.3399810435848560
      z(2) =.8611363115940530
c
      w(1) =.6521451548625460
      w(2) =.3478548451374540
c
      return
      end
```

```
      subroutine gaus4(z,w)
      real z(24),w(24)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine gaus4 contains the roots of  8th order                  c
c   Legendre polynomial                                                c
c                                                                      c
c   called by: various integration routines prefixed gq                c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      z(1) =.1834346424956500
      z(2) =.5255324099163290
      z(3) =.7966664774136270
      z(4) =.9602898564975360
c
      w(1) =.3626837833783620
      w(2) =.3137066458778870
      w(3) =.2223810344533740
      w(4) =.1012285362903760
c
      return
      end
```

```
      subroutine gaus6(z,w)
      real z(24),w(24)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine gaus6 contains the roots of 12th order                 c
c   Legendre polynomial                                               c
c                                                                     c
c   called by: various integration routines prefixed gq               c
c                                                                     c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      z(1) =.125233408511469
      z(2) =.367831498998180
      z(3) =.587317954286617
      z(4) =.769902674194305
      z(5) =.904117256370475
      z(6) =.981560634246719
c
      w(1) =.249147045813403
      w(2) =.233492536538355
      w(3) =.203167426723066
      w(4) =.160078328543346
      w(5) =.106939325995318
      w(6) =.047175336386512
c
      return
      end
```

```
      subroutine gaus24(z,w)
      real z(24),w(24)
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Subroutine gaus24 contains the roots of 48th order                 c
c   Legendre polynomial                                                c
c                                                                      c
c   called by: various integration routines prefixed gq                c
c                                                                      c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      z(1) =.03238017096286936203
      z(2) =.09700469920946269893
      z(3) =.16122355606889171805
      z(4) =.22476379039468906122
      z(5) =.28736248735545557673
      z(6) =.34875588629216073816
      z(7) =.40868648199071672991
      z(8) =.46690290475095840454
      z(9) =.52316097472223303367
      z(10)=.57722472608397270381
      z(11)=.62868739677651362399
      z(12)=.67787237963266390521
      z(13)=.72403413092381465467
      z(14)=.76715903251574033925
      z(15)=.80706620402944262708
      z(16)=.84358826162439353071
      z(17)=.87657202027424788590
      z(18)=.90587913671556967282
      z(19)=.93138669070655433311
      z(20)=.95298770316043086072
      z(21)=.97059159254624725046
      z(22)=.98412458372282685774
      z(23)=.99353017226635075754
      z(24)=.99877100725242611860
c
      w(1) =.06473769681268392250
      w(2) =.06446616443595008220
      w(3) =.06392423858464818662
      w(4) =.06311419228625402566
      w(5) =.06203942315989266390
      w(6) =.06070443916589388005
      w(7) =.05911483969839563574
      w(8) =.05727729210040321570
      w(9) =.05519950369998416286
      w(10)=.05289018948519366709
      w(11)=.05035903555385447495
      w(12)=.04761665849249047482
      w(13)=.04467456085669428041
      w(14)=.04154508294346474921
      w(15)=.03824135106583070631
      w(16)=.03477722256477043889
      w(17)=.03116722783279808890
      w(18)=.02742650970835694820
      w(19)=.02357076083932437914
      w(20)=.01961616045735552781
      w(21)=.01557931572294384872
      w(22)=.01147723457923453949
      w(23)=.00732755390127626210
      w(24)=.00315334605230583863
c
      return
      end
```

```
      program org main
      real* 8 lngth,wdth,wf,wq,hf,
     +        xs0,wsc,lsc,t,er,
     +        lx,ly,lz,freq,xd0,flang
      integer elnumc,ncl,ncw,m1,flgair,flgdip,flgms,
     +        nf,mi,elnumd,md1,md2,nld,nhd,nwd
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Program org organizes the output data for further analysis      c
c   The current vector is written to the file 'cur', other data     c
c   necessary for the analysis is written to the file 'dat'         c
c                                                                   c
c   calls: orgsub                                                   c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      read(5,*)
      read(5,*)
      read(5,*)
      read(5,*) flgair,flgdip,flgms
      read(5,*) ncl,ncw,lngth,wdth,wf,m1,elnumc
      read(5,*) xs0,wsc,lsc,xd0,mi,flang
      read(5,*) elnumd,md1,md2,nld,nhd,nwd,t
      read(5,*) lx,ly,lz,freq,er
      read(5,*) wq,hf,nf
      nn=elnumc+elnumd
      call orgsub(ncl,ncw,lngth,wdth,wf,m1,elnumc,xs0,lsc,
     +            wsc,xd0,elnumd,flgair,flgdip,flgms,flang,
     +            md1,md2,nld,nhd,nwd,t,lx,ly,lz,freq,er,nn,
     +            wq,hf,nf,mi)
      end
c
      subroutine orgsub(ncl,ncw,lngth,wdth,wf,m1,elnumc,xs0,lsc,
     +            wsc,xd0,elnumd,flgair,flgdip,flgms,flang,
     +            md1,md2,nld,nhd,nwd,t,lx,ly,lz,freq,er,nn,
     +            wq,hf,nf,mi)
      real* 8 lngth,wdth,h,w,wf,pi,teta,phi,eabs(200),max,wq,
     +        hu(2000),wu(2000),xs0,wsc,lsc,tetdum,tc,t,er,hf,
     +        lx,ly,lz,klx,kly,klz,kwf,freq,xd0,kt,kxs0,flang
      complex* 16 r(nn),eteta,p1,zs
      integer i,j,cut,no,elnumc,dum2(2000),ncl,ncw,m1,dum(2000),
     +        mi,elnumd,md1,md2,nld,nhd,nwd,nn,flgair,flgdip,
     +        flgms,nf
      read(5,*) dum2,dum,wu,hu
c
      read(5,*) r
      open (unit=9,file='dat')
      open (unit=2,file='cur')
c
      write(9,*) flgair,flgdip,flgms
      write(9,*) ncl,ncw,lngth,wdth,wf,m1,elnumc
      write(9,*) xs0,wsc,lsc,xd0,mi,flang
      write(9,*) elnumd,md1,md2,nld,nhd,nwd,t
      write(9,*) lx,ly,lz,freq,er
      write(9,*) wq,hf,nf
      do 175 i=1,4*m1+nf
      if (i.le.m1) then
      write(9,*) dum2(i),dum(i),wu(i),hu(i)
      else
      write(9,*) dum2(i)
      endif
  175 continue
      do 176 i=1,nn
      write(2,*) r(i)
  176 continue
      close(unit=9)
```

```
close(unit=2)
return
end
```

```
      program pattern main
      real* 8 lngth,wdth,h,w,wf,pi,teta,phi,max,wq,hf,z0,
     +        hu(2000),wu(2000),xs0,wsc,lsc,tetdum,tc,t,er,
     +        lx,ly,lz,klx,kly,klz,kwf,freq,xd0,kt,kxs0,flang,
     +        eabste(901),eabspe(901),eabsth(901),eabsph(901)
      complex* 16 r(5000),eteta,pi,zsteta,zsphi,ephi
      integer i,j,cut,no,elnumc,dum2(2000),ncl,ncw,m1,dum(2000),
     +        mi,elnumd,md1,md2,nld,nhd,nwd,flgair,flgdip,
     +        flgms,nf,flgxp
      character cur*15,dat*15,hacp*15,eacp*15,haxp*15,eaxp*15
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Program pattern  computes the far-fields of the antenna        c
c   in the E and H planes of the antenna only                      c
c                                                                  c
c   calls: fsource, farfld, ediel, calcc, dipfld                   c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      write(6,*) 'enter current and data file name'
      read(5,*) cur,dat
      write(6,*) 'enter ecp,hcp,exp,hxp'
      read(5,*) eacp,hacp,eaxp,haxp
      flgx=0
      write(6,*) 'enter 1 if x-pol is required'
      write(6,*) 'note: for air LTSA x-pol shouldn't be requested'
      write(6,*) 'since it is zero, underflow may result !'
      open (unit=2,file=cur)
      open (unit=9,file=dat)
      open (unit=3,file=hacp)
      open (unit=7,file=eacp)
      if (flgx.eq.1) then
      open (unit=33,file=haxp)
      open (unit=77,file=eaxp)
      endif
      pi=datan(1.d0)*4.d0
      p1=(0.d0,1.d0)
      read(9,*) flgair,flgdip,flgms
      read(9,*) ncl,ncw,lngth,wdth,wf,m1,elnumc
      read(9,*) xs0,wsc,lsc,xd0,mi,flang
      read(9,*) elnumd,md1,md2,nld,nhd,nwd,t
      read(9,*) lx,ly,lz,freq,er
      read(9,*) wq,hf,nf
      klx=2.d0*pi*lx
      kly=2.d0*pi*ly
      klz=2.d0*pi*lz
      kt=2.d0*pi*t
      kwf=2.d0*pi*wf
      kxs0=2.d0*pi*xs0
      z0=wdth*2.d0*pi+kwf/2.d0
      do 7 j=1,4*m1+nf
      if (j.le.m1) then
      read(9,*) dum2(j),dum(j),wu(j),hu(j)
      else
      read(9,*) dum2(j)
      endif
    7 continue
      w=lngth/dble(ncl)
      h=wdth/dble(ncw)
c
      do 10 i=1,elnumc+elnumd
      read(2,*) r(i)
   10 continue
      max=0.d0
```

```
      do 835 cut=0,1
      if (cut.eq.1) then
      teta=90.d0
      teta=teta*pi/180.d0
      phi=0.d0
      else
      phi=0.d0
      phi=phi*pi/180.d0
      teta=0.d0
      endif
c   no=number of data points
      no=360
      tc=2.d0*pi/dble(no)
c
      do 20 j=0,no
      if (cut.eq.1) then
      phi=tc*dble(j)
      tetdum=teta
      else
      teta=tc*dble(j)
      if (teta.gt.pi) then
      tetdum=2.d0*pi-teta
      phi=pi
      else
      tetdum=teta
      phi=0.d0
      endif
      endif
c
c   call the far-field calculating routine
c
      call farfld(ncl,dum2,wdth,r,h,w,tetdum,phi,eteta,m1,pi,wf,
     +            wu,hu,xs0,wsc,lsc,t,er,mi,ephi,xd0,
     +            md1,md2,nld,klx,kly,klz,elnumc,elnumd,wq,hf,nf)
      if (flgdip.eq.1) goto 892
c
c   add the far-field of the source
c
      call fsource(p1,pi,tetdum,phi,kwf,kxs0,kt,zsteta,zsphi,
     +             flgms,z0)
      eteta=eteta-zsteta
      ephi=ephi-zsphi
 892  continue
c
c   find the max of the field and organize
c
      if (cut.eq.1) then
      eabsth(j+1)=cdabs(eteta)
      if (eabsth(j+1).gt.max) max=eabsth(j+1)
      if (flgx.eq.1) then
      eabsph(j+1)=cdabs(ephi)
      if (eabsph(j+1).gt.max) max=eabsph(j+1)
      endif
      else
      eabste(j+1)=cdabs(eteta)
      if (eabste(j+1).gt.max) max=eabste(j+1)
      if (flgx.eq.1) then
      eabspe(j+1)=cdabs(ephi)
      if (eabspe(j+1).gt.max) max=eabspe(j+1)
      endif
      endif
  20  continue
 835  continue
c
c   calculate dB values and write to output files
c
      do 836 cut=0,1
      do 30 j=0,no
```

```
      if (cut.eq.1) then
      eabsth(j+1)=20.d0*dlog(eabsth(j+1)/max)/dlog(10.d0)
      if (flgx.eq.1) then
      eabsph(j+1)=20.d0*dlog(eabsph(j+1)/max)/dlog(10.d0)
      endif
      else
      eabste(j+1)=20.d0*dlog(eabste(j+1)/max)/dlog(10.d0)
      if (flgx.eq.1) then
      eabspe(j+1)=20.d0*dlog(eabspe(j+1)/max)/dlog(10.d0)
      endif
      endif
      if (cut.ne.1) then
      phi=tc*dble(j)*180.d0/pi
      else
      phi=tc*dble(j)*180.d0/pi
      endif
      if (cut.eq.1) then
      write(3,*) phi,eabsth(j+1)
      if (flgx.eq.1) then
      write(33,*) phi,eabsph(j+1)
      endif
      else
      write(7,*) phi,eabste(j+1)
      if (flgx.eq.1) then
      write(77,*) phi,eabspe(j+1)
      endif
      endif
   30 continue
  836 continue
      end
c
      subroutine fsource(p1,pi,teta,phi,kwf,kxs0,kt,
     +                   zsteta,zsphi,flgms,z0)
      real* 8 pi,teta,phi,kwf,kxs0,kt,z0
      complex* 16 zs,p1,zsteta,zsphi
      integer flgms
c
c   subroutine fsource calculates the far-fields of the source
c
      if (flgms.eq.0) then
      zs=kwf*dsin(teta)
      zs=zs*cdexp(p1*kxs0*dsin(teta)*dcos(phi))
      zsteta=zs*cdexp(p1*kt*dsin(teta)*dsin(phi))
      zsphi=0.d0
      else
      zs=kt*cdexp(p1*kxs0*dsin(teta)*dcos(phi))
      zs=zs*cdexp(p1*kt/2.d0*dsin(teta)*dsin(phi))
      zs=zs*cdexp(-p1*z0*dcos(teta))
      zsteta=-zs*dcos(teta)*dsin(phi)
      zsphi=-zs*dcos(phi)
      endif
      return
      end
c
      subroutine farfld(ncl,dum2,wdth,r,h,w,teta,phi,eteta,m1,pi,
     +                  wf,wu,hu,xs0,wsc,lsc,t,er,mi,ephi,xd0,
     +            md1,md2,nld,klx,kly,klz,elnumc,elnumd,wq,hf,nf)
      real* 8 h1,w1,h,w,teta,phi,gam,alp,y1,x1,wdth,pi,d,hh,wf,
     +        i1,i2,wu(2000),hu(2000),xs0,wsc,lsc,t,er,l,xd0,
     +        hfi,hi1,h21,wq,hf,
     +        klx,kly,klz
      complex* 16 r(5000),eteta,ephi,z12,r1,r2,r3,a,b
      integer i,j,m1,ip,ii,jj,dum2(2000),ncl,mi,
     +        md1,md2,nld,elnumc,elnumd,aj,it,ih,iw,il,nf
c
```

```
c   subroutine farfld calculates the far-fields of the antenna
c
      eteta=0.d0
      ephi=0.d0
c
      do 10 i=1,4*m1+nf+md2
      if (i.le.(4*m1+nf)) then
      if (dum2(i).eq.0) goto 551
      z12=0.d0
      do 23 j=1,2
      if (j.eq.1) then
      i1=0.d0
      i2=1.d0
      else
      i1=1.d0
      i2=0.d0
      endif
c
      if ((i.ge.0).and.(i.le.m1)) then
      gam=pi/2.d0
      alp=1.5d0*pi
      ii=int((i-1)/ncl)+1
      jj=i-(ii-1)*ncl
      if (j.eq.1) then
      w1=hu(i+1)/2.d0
      h1=wu(i)
      y1=dble(jj-1)*w
      x1=wdth-dble(ii-1)*h-hu(i+1)/2.d0+wf/2.d0
      else
      w1=hu(i+1)/2.d0
      h1=wu(i+1)
      y1=dble(jj)*w
      x1=wdth-dble(ii-1)*h-hu(i+1)/2.d0+wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.m1).and.(i.le.(2*m1))) then
      gam=pi/2.d0
      alp=1.5d0*pi
      ii=int((i-m1-1)/ncl)+1
      jj=i-m1-(ii-1)*ncl
      ip=i-(2*ii-1)*ncl
      if (j.eq.1) then
      w1=hu(ip+1)/2.d0
      h1=wu(ip)
      y1=dble(jj-1)*w
      x1=-dble(ii)*h+hu(ip+1)/2.d0-wf/2.d0
      else
      w1=hu(ip+1)/2.d0
      h1=wu(ip+1)
      y1=dble(jj)*w
      x1=-dble(ii)*h+hu(ip+1)/2.d0-wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(2*m1)).and.(i.le.(3*m1))) then
      gam=0.d0
      alp=0.d0
      ii=int((i-2*m1-1)/ncl)+1
      jj=i-2*m1-(ii-1)*ncl
      ip=i-2*m1
      if (j.eq.1) then
      w1=wu(ip)/2.d0
      h1=hu(ip)
```

```
      x1=dble(jj-1)*w+wu(ip)/2.d0
      y1=dble(ii-1)*h-wdth-wf/2.d0
      else
      w1=wu(ip+ncl)/2.d0
      h1=hu(ip+ncl)
      x1=dble(jj-1)*w+wu(ip+ncl)/2.d0
      y1=dble(ii)*h-wdth-wf/2.d0
      endif
      l=0.d0
      endif
c
      if (i.eq.(4*m1+i)) then
      gam=0.d0
      alp=0.d0
      w1=wsc
      h1=lsc
      if (j.eq.1) then
      x1=w1
      y1=-h1
      else
      x1=w1
      y1=0.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(3*m1)).and.(i.le.(4*m1))) then
      gam=0.d0
      alp=0.d0
      ii=int((i-3*m1-1)/ncl)+1
      jj=i-3*m1-(ii-1)*ncl
      ip=i-(2*ii-1)*ncl-2*m1
      if (j.eq.1) then
      w1=wu(ip)/2.d0
      h1=hu(ip)
      x1=dble((jj-1))*w+wu(ip)/2.d0
      y1=dble(ii)*h-hu(ip)+wf/2.d0
      else
      w1=wu(ip-ncl)/2.d0
      h1=hu(ip-ncl)
      x1=dble((jj-1))*w+wu(ip-ncl)/2.d0
      y1=dble(ii)*h+wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(4*m1+1)).and.(i.le.(4*m1+nf))) then
      gam=0.d0
      alp=0.d0
      ii=i-4*m1-1
      w1=wq/2.d0
      h1=hf
      if (j.eq.1) then
      x1=xs0
      y1=dble(ii-1)*hf-wdth-wf/2.d0
      else
      x1=xs0
      y1=dble(ii)*hf-wdth-wf/2.d0
      endif
      l=t
      endif
c
c   calculate the distance of the monopole from the origin
c
      d=dcos(alp)*x1-dsin(alp)*y1
      hh=dsin(alp)*x1+dcos(alp)*y1
      call dipfld(r(dum2(i)),h1,w1,d,hh,l,teta,phi,gam,alp,
```

```
     +            pi,a,b,i1,i2)
      eteta=eteta+a
      ephi=ephi+b
   23 continue
      else
c
c   calculate the distance of dielectric current from the origin
c
      aj=i-4*m1-nf
      it=int((aj-1)/md2)
      ih=int((aj-it*md2-1)/md1)
      iw=int((aj-it*md2-ih*md1-1)/nld)
      il=aj-it*md2-ih*md1-iw*nld
      r1=r(elnumc+aj)
      r2=r(elnumc+aj+md2)
      r3=r(elnumc+aj+2*md2)
      d=xd0+dble(il)*klx-klx/2.d0
      hh=dble(ih)*kly+kly/2.d0
      l=dble(iw)*klz+klz/2.d0-(wdth+wf/2.d0)*2.d0*pi
      call ediel(teta,phi,d,hh,l,r1,r2,r3,klx,kly,klz,a,b)
      eteta=eteta+a
      ephi=ephi+b
      endif
  551 continue
   10 continue
      return
      end
c
      subroutine ediel(teta,phi,d,hh,l,r1,r2,r3,klx,kly,klz,a,b)
      real* 8 teta,phi,d,hh,l,klx,kly,klz
      complex* 16 r,a,c,b,r1,r2,r3
c
c   subroutine ediel calculates far-fields of dielectric part
c   calls: calcc for the calculation of a common term
c
      call calcc(teta,phi,klx,kly,klz,d,hh,l,c)
      a=c*(dcos(teta)*dcos(phi)*r1*klx+dcos(teta)*dsin(phi)*r2*kly-
     +     dsin(teta)*r3*klz)
      b=c*(-dsin(phi)*r1*klx+dcos(phi)*r2*kly)
      return
      end
c
      subroutine calcc(teta,phi,klx,kly,klz,d,hh,l,c)
      real* 8 teta,phi,klx,kly,klz,d,hh,l,c1,c2,c3,c4,c5
      complex* 16 c,p1
c
c   subroutine calcc calculates the common term in dielectric
c   far-field calculations
c
      p1=(0.d0,1.d0)
      c1=d*dsin(teta)*dcos(phi)+hh*dsin(teta)*dsin(phi)+l*dcos(teta)
      c2=klx*dsin(teta)*dcos(phi)/2.d0
      c3=1.d0
      if (c2.ne.(0.d0)) c3=dsin(c2)/c2
      c2=kly*dsin(teta)*dsin(phi)/2.d0
      c4=1.d0
      if (c2.ne.(0.d0)) c4=dsin(c2)/c2
      c2=klz*dcos(teta)/2.d0
      c5=1.d0
      if (c2.ne.(0.d0)) c5=dsin(c2)/c2
      c=cdexp(p1*c1)*c3*c4*c5
      return
      end
c
      subroutine dipfld(r,h1,w1,d,hh,l,teta,phi,gam,alp,
     +                  pi,ete,eph,i1,i2)
      real* 8 h1,w1,d,hh,teta,phi,gam,alp,pi,c,b,a,i1,i2,a2,kh1,l
      complex* 16 ete,aa,p1,r,c2,q1,q2,eph
c
```

```
c   subroutine dipfld calculates the far-fields of the monopole
c   currents for the conducting parts of the antenna using a
c   closed form formula obtained from the vector potential
c   formulation
c
      kh1=2.d0*pi*h1
      c=(d*dsin(teta)*dcos(phi)+hh*dcos(teta)+
     +   l*dsin(teta)*dsin(phi))*2.d0*pi
      b=(dcos(alp)*dsin(teta)*dcos(phi)+dsin(alp)*dcos(teta))*2.d0*pi
      a=dcos(alp)*dcos(teta)-dsin(alp)*dsin(teta)*dcos(phi)
      p1=(0.d0,1.d0)
      a2=2.d0*pi*a
      q1=i1*(dcos(a2*h1)-dcos(kh1)+p1*dsin(a2*h1)-p1*a*dsin(kh1))
      q2=cdexp(p1*a2*h1)*(p1*a*dsin(kh1)-dcos(kh1))
      if ((a.ne.(-1.d0)).and.(a.ne.(1.d0))) then
      q1=q1/((1.d0-a**2)*dsin(kh1))
      q2=i2*(q2+1.d0)/((1.d0-a**2)*dsin(kh1))
      else
      q1=-kh1*dsin(a*kh1)+p1*kh1*dcos(a*kh1)-p1*dsin(kh1)
      q1=i1*q1/(-2.d0*a*dsin(kh1))
      q2=-kh1*a*dsin(kh1)-p1*kh1*dcos(kh1)+p1*dsin(kh1)
      q2=i2*q2*cdexp(p1*a*kh1)/(-2.d0*a*dsin(kh1))
      endif
      aa=cdexp(p1*c)
      c2=1.d0
      if ((b*w1).ne.(0.d0)) then
      c2=c2*dsin(w1*b)/(w1*b)
      endif
      aa=aa*c2*(q1+q2)*r
      ete=aa*(-dcos(teta)*dcos(phi)*dsin(alp)-dcos(alp)*dsin(teta))
69    continue
      eph=aa*(dsin(phi)*dsin(alp))
      return
      end
```

```
      program patdd main
      real* 8 lngth,wdth,h,w,wf,pi,teta,phi,max,wq,hf,
     +        hu(2000),wu(2000),xs0,wsc,lsc,tc,t,er,
     +        lx,ly,lz,klx,kly,klz,kwf,freq,xd0,kt,kxs0,flang,
     +        dcpabs(901),dxpabs(901),phim,beta,gam,z0
      complex* 16 r(5000),eteta,p1,zsteta,zsphi,ephi,edcp,edxp
      integer i,j,cut,no,elnumc,dum2(2000),ncl,ncw,m1,dum(2000),
     +        mi,elnumd,md1,md2,nld,nhd,nwd,flgair,flgdip,
     +        flgms,nf
      character cur*15,dat*15,dcp*15,dxp*15
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c   Program patdd  computes the far-fields of the antenna          c
c   in a plane specified with the angles beta and gamma            c
c                                                                  c
c   beta: offset angle of TSA from the rotation axis               c
c   gamma: offset angle of standard gain antenna from rot. axis    c
c                                                                  c
c                                                                  c
c   calls: fsource, farfld, ediel, calcc, dipfld                   c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      write(6,*) 'enter current and data file name'
      read(5,*) cur,dat
      write(6,*) 'enter dcp,dxp'
      read(5,*) dcp,dxp
      open (unit=2,file=cur)
      open (unit=9,file=dat)
      open (unit=3,file=dcp)
      open (unit=7,file=dxp)
      pi=datan(1.d0)*4.d0
      p1=(0.d0,1.d0)
      read(9,*) flgair,flgdip,flgms
      read(9,*) ncl,ncw,lngth,wdth,wf,m1,elnumc
      read(9,*) xs0,wsc,lsc,xd0,mi,flang
      read(9,*) elnumd,md1,md2,nld,nhd,nwd,t
      read(9,*) lx,ly,lz,freq,er
      read(9,*) wq,hf,nf
      klx=2.d0*pi*lx
      kly=2.d0*pi*ly
      klz=2.d0*pi*lz
      kt=2.d0*pi*t
      kwf=2.d0*pi*wf
      kxs0=2.d0*pi*xs0
      z0=wdth*2.d0*pi+kwf/2.d0
      do 7 j=1,4*m1+nf
      if (j.le.m1) then
      read(9,*) dum2(j),dum(j),wu(j),hu(j)
      else
      read(9,*) dum2(j)
      endif
    7 continue
      w=lngth/dble(ncl)
      h=wdth/dble(ncw)
c
      do 10 i=1,elnumc+elnumd
      read(2,*) r(i)
   10 continue
      max=0.d0
      write(6,*) ' Enter beta-offset angle of TSA'
      read(5,*) beta
      write(6,*) ' Enter gamma-offset angle of measurement ant. '
      read(5,*) gam
      beta=beta*pi/180.d0
```

```
      gam=gam*pi/180.d0
      no=360
      tc=2.d0*pi/dble(no)
c
      do 20 j=0,no
      phim=tc*dble(j)
c
c   calculate the pattern angles as seen by LTSA
c
      call calc(pi,phim,beta,teta,phi)
c
c   call the far-field calculating routine
c
      call farfld(ncl,dum2,wdth,r,h,w,teta,phi,eteta,mi,pi,wf,
     +            wu,hu,xs0,wsc,lsc,t,er,mi,ephi,xd0,
     +            md1,md2,nld,klx,kly,klz,elnumc,elnumd,wq,hf,nf)
  861 continue
      if (flgdip.eq.1) goto 892
c
c   add the far-field of the source
c
      call fsource(p1,pi,teta,phi,kwf,kxs0,kt,zsteta,zsphi,
     +             flgms,z0)
      eteta=eteta-zsteta
      ephi=ephi-zsphi
  892 continue
c
c   calculate the measured quantity in terms of theta and phi
c   components of the electric field intensity of LTSA
c
      call meas(pi,phim,beta,gam,teta,phi,eteta,ephi,edcp,edxp)
c
c   find the max of the field and organize
c
      dcpabs(j+1)=cdabs(edcp)
      dxpabs(j+1)=cdabs(edxp)
      if (dcpabs(j+1).gt.max) max=dcpabs(j+1)
      if (dxpabs(j+1).gt.max) max=dxpabs(j+1)
   20 continue
c
c   calculate dB values and write to output files
c
      do 30 j=0,no
      dcpabs(j+1)=20.d0*dlog(dcpabs(j+1)/max)/dlog(10.d0)
      dxpabs(j+1)=20.d0*dlog(dxpabs(j+1)/max)/dlog(10.d0)
      phi=tc*dble(j)*180.d0/pi
      if (phi.gt.(180.d0)) phi=phi-360.d0
      write(3,*) phi,dcpabs(j+1)
      write(7,*) phi,dxpabs(j+1)
   30 continue
      end
c
      subroutine meas(pi,phim,beta,gam,teta,phi,
     +                eteta,ephi,edcp,edxp)
      real* 8 pi,phim,beta,gam,teta,phi,a,b,c,d,gam2
      complex* 16 eteta,ephi,edcp,edxp
c
c   subroutine meas
c   calculates the measured quantity in terms of theta and phi
c   components of the electric field intensity of LTSA
c
      gam2=gam+pi/2.d0
      a=-dsin(phim)*dcos(teta)*dcos(phi)
      a=a+dcos(phim)*dcos(beta)*dcos(teta)*dsin(phi)
      a=a-dcos(phim)*dsin(beta)*dsin(teta)
      b=dsin(phim)*dsin(phi)+dcos(phim)*dcos(beta)*dcos(phi)
      c=dsin(beta)*dcos(teta)*dsin(phi)+dcos(beta)*dsin(teta)
      d=dsin(beta)*dcos(phi)
```

```
      edcp=(a*dsin(gam)-c*dcos(gam))*eteta
      edcp=edcp+(b*dsin(gam)-d*dcos(gam))*ephi
      edxp=(a*dsin(gam2)-c*dcos(gam2))*eteta
      edxp=edxp+(b*dsin(gam2)-d*dcos(gam2))*ephi
      return
      end
c
      subroutine calc(pi,phim,beta,teta,phi)
      real* 8 phim,beta,teta,phi,x,y,z,pi
c
c     subroutine calc calculates the pattern angles as seen by LTSA
c
      x=dcos(phim)
      y=dsin(phim)*dcos(beta)
      z=dsin(phim)*dsin(beta)
c
      if ((beta.eq.(pi/2.d0)).and.(y.lt.(1.d-7))) then
      if (x.ge.(0.d0)) phi=0.d0
      if (x.lt.(0.d0)) phi=180.d0
      else
      if (x.eq.(0.d0)) then
      if (y.gt.(0.d0)) phi=pi/2.d0
      if (y.lt.(0.d0)) phi=3.d0*pi/2.d0
      else
      if ((y/x).lt.(0.d0)) then
      if (y.lt.(0.d0)) phi=datan(y/x)+pi*2.d0
      if (x.lt.(0.d0)) phi=datan(y/x)+pi
      else
      if ((y/x).ge.(0.d0)) then
      if (y.lt.(0.d0)) phi=datan(y/x)+pi
      if (y.gt.(0.d0)) phi=datan(y/x)
      endif
      endif
      endif
      endif
      teta=dacos(z)
      return
      end
c
      subroutine fsource(p1,pi,teta,phi,kwf,kxs0,kt,
     +                   zsteta,zsphi,flgms,z0)
      real* 8 pi,teta,phi,kwf,kxs0,kt,z0
      complex* 16 zs,p1,zsteta,zsphi
      integer flgms
c
c     subroutine fsource calculates the far-fields of the source
c
      if (flgms.eq.0) then
      zs=kwf*dsin(teta)
      zs=zs*cdexp(p1*kxs0*dsin(teta)*dcos(phi))
      zsteta=zs*cdexp(p1*kt*dsin(teta)*dsin(phi))
      zsphi=0.d0
      else
      zs=kt*cdexp(p1*kxs0*dsin(teta)*dcos(phi))
      zs=zs*cdexp(p1*kt/2.d0*dsin(teta)*dsin(phi))
      zs=zs*cdexp(-p1*z0*dcos(teta))
      zsteta=-zs*dcos(teta)*dsin(phi)
      zsphi=-zs*dcos(phi)
      endif
      return
      end
c
      subroutine farfld(ncl,dum2,wdth,r,h,w,teta,phi,eteta,m1,pi,
     +                  wf,wu,hu,xs0,wsc,lsc,t,er,mi,ephi,xd0,
     +            md1,md2,nld,klx,kly,klz,elnumc,elnumd,wq,hf,nf)
      real* 8 hi,w1,h,w,teta,phi,gam,alp,y1,x1,wdth,pi,d,hh,wf,
```

```
     +        i1,i2,wu(2000),hu(2000),xs0,wsc,lsc,t,er,l,xd0,
     +        hfi,hi1,h21,klx,kly,klz,wq,hf
      complex* 16 r(5000),eteta,ephi,z12,r1,r2,r3,a,b
      integer i,j,m1,ip,ii,jj,dum2(2000),ncl,mi,nf,
     +        md1,md2,nld,elnumc,elnumd,aj,it,ih,iw,il
c
cc    subroutine farfld calculates the far-fields of the antenna
c
      eteta=0.d0
      ephi=0.d0
c
      do 10 i=1,4*m1+nf+md2
      if (i.le.(4*m1+nf)) then
      if (dum2(i).eq.0) goto 551
      z12=0.d0
      do 23 j=1,2
      if (j.eq.i) then
      i1=0.d0
      i2=1.d0
      else
      i1=1.d0
      i2=0.d0
      endif
c
      if ((i.ge.0).and.(i.le.m1)) then
      gam=pi/2.d0
      alp=1.5d0*pi
      ii=int((i-1)/ncl)+1
      jj=i-(ii-1)*ncl
      if (j.eq.1) then
      w1=hu(i+1)/2.d0
      h1=wu(i)
      y1=dble(jj-1)*w
      x1=wdth-dble(ii-1)*h-hu(i+1)/2.d0+wf/2.d0
      else
      w1=hu(i+1)/2.d0
      h1=wu(i+1)
      y1=dble(jj)*w
      x1=wdth-dble(ii-1)*h-hu(i+1)/2.d0+wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.m1).and.(i.le.(2*m1))) then
      gam=pi/2.d0
      alp=1.5d0*pi
      ii=int((i-m1-1)/ncl)+1
      jj=i-m1-(ii-1)*ncl
      ip=i-(2*ii-1)*ncl
      if (j.eq.1) then
      w1=hu(ip+1)/2.d0
      h1=wu(ip)
      y1=dble(jj-1)*w
      x1=-dble(ii)*h+hu(ip+1)/2.d0-wf/2.d0
      else
      w1=hu(ip+1)/2.d0
      h1=wu(ip+1)
      y1=dble(jj)*w
      x1=-dble(ii)*h+hu(ip+1)/2.d0-wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(2*m1)).and.(i.le.(3*m1))) then
      gam=0.d0
      alp=0.d0
```

```
      ii=int((i-2*m1-1)/ncl)+1
      jj=i-2*m1-(ii-1)*ncl
      ip=i-2*m1
      if (j.eq.1) then
      w1=wu(ip)/2.d0
      h1=hu(ip)
      x1=dble(jj-1)*w+wu(ip)/2.d0
      y1=dble(ii-1)*h-wdth-wf/2.d0
      else
      w1=wu(ip+ncl)/2.d0
      h1=hu(ip+ncl)
      x1=dble(jj-1)*w+wu(ip+ncl)/2.d0
      y1=dble(ii)*h-wdth-wf/2.d0
      endif
      l=0.d0
      endif
c
      if (i.eq.(4*m1+1)) then
      gam=0.d0
      alp=0.d0
      w1=wsc
      h1=lsc
      if (j.eq.1) then
      x1=w1
      y1=-h1
      else
      x1=w1
      y1=0.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(3*m1)).and.(i.le.(4*m1))) then
      gam=0.d0
      alp=0.d0
      ii=int((i-3*m1-1)/ncl)+1
      jj=i-3*m1-(ii-1)*ncl
      ip=i-(2*ii-1)*ncl-2*m1
      if (j.eq.1) then
      w1=wu(ip)/2.d0
      h1=hu(ip)
      x1=dble((jj-1))*w+wu(ip)/2.d0
      y1=dble(ii)*h-hu(ip)+wf/2.d0
      else
      w1=wu(ip-ncl)/2.d0
      h1=hu(ip-ncl)
      x1=dble((jj-1))*w+wu(ip-ncl)/2.d0
      y1=dble(ii)*h+wf/2.d0
      endif
      l=0.d0
      endif
c
      if ((i.gt.(4*m1+1)).and.(i.le.(4*m1+nf))) then
      gam=0.d0
      alp=0.d0
      ii=i-4*m1-1
      w1=wq/2.d0
      h1=hf
      if (j.eq.1) then
      x1=xs0
      y1=dble(ii-1)*hf-wdth-wf/2.d0
      else
      x1=xs0
      y1=dble(ii)*hf-wdth-wf/2.d0
      endif
      l=t
```

```
      endif
c
c   calculate the distance of the monopole from the origin
c
      d=dcos(alp)*x1-dsin(alp)*y1
      hh=dsin(alp)*x1+dcos(alp)*y1
      call dipfld(r(dum2(i)),h1,w1,d,hh,l,teta,phi,gam,alp,
     +            pi,a,b,i1,i2)
      eteta=eteta+a
      ephi=ephi+b
   23 continue
      else
c
c   calculate the distance of dielectric current from the origin
c
      aj=i-4*m1-nf
      it=int((aj-1)/md2)
      ih=int((aj-it*md2-1)/md1)
      iw=int((aj-it*md2-ih*md1-1)/nld)
      il=aj-it*md2-ih*md1-iw*nld
      r1=r(elnumc+aj)
      r2=r(elnumc+aj+md2)
      r3=r(elnumc+aj+2*md2)
      d=xd0+dble(il)*klx-klx/2.d0
      hh=dble(ih)*kly+kly/2.d0
      l=dble(iw)*klz+klz/2.d0-(wdth+wf/2.d0)*2.d0*pi
      call ediel(teta,phi,d,hh,l,r1,r2,r3,klx,kly,klz,a,b)
      eteta=eteta+a
      ephi=ephi+b
      endif
  551 continue
   10 continue
      return
      end
c
      subroutine ediel(teta,phi,d,hh,l,r1,r2,r3,klx,kly,klz,a,b)
      real* 8 teta,phi,d,hh,l,klx,kly,klz
      complex* 16 r,a,c,b,r1,r2,r3
c
c   subroutine ediel calculates far-fields of dielectric part
c   calls: calcc for the calculation of a common term
c
      call calcc(teta,phi,klx,kly,klz,d,hh,l,c)
      a=c*(dcos(teta)*dcos(phi)*r1*klx+dcos(teta)*dsin(phi)*r2*kly-
     +     dsin(teta)*r3*klz)
      b=c*(-dsin(phi)*r1*klx+dcos(phi)*r2*kly)
      return
      end
c
      subroutine calcc(teta,phi,klx,kly,klz,d,hh,l,c)
      real* 8 teta,phi,klx,kly,klz,d,hh,l,c1,c2,c3,c4,c5
      complex* 16 c,p1
c
c   subroutine calcc calculates the common term in dielectric
c   far-field calculations
c
      p1=(0.d0,1.d0)
      c1=d*dsin(teta)*dcos(phi)+hh*dsin(teta)*dsin(phi)+l*dcos(teta)
      c2=klx*dsin(teta)*dcos(phi)/2.d0
      c3=1.d0
      if (c2.ne.(0.d0)) c3=dsin(c2)/c2
      c2=kly*dsin(teta)*dsin(phi)/2.d0
      c4=1.d0
      if (c2.ne.(0.d0)) c4=dsin(c2)/c2
      c2=klz*dcos(teta)/2.d0
      c5=1.d0
      if (c2.ne.(0.d0)) c5=dsin(c2)/c2
      c=cdexp(p1*c1)*c3*c4*c5
      return
```

```
      end
c
      subroutine dipfld(r,h1,w1,d,hh,l,teta,phi,gam,alp,
     +                  pi,ete,eph,i1,i2)
      real* 8 h1,w1,d,hh,teta,phi,gam,alp,pi,c,b,a,i1,i2,a2,kh1,l
      complex* 16 ete,aa,p1,r,c2,q1,q2,eph
c
c     subroutine dipfld calculates the far-fields of the monopole
c     currents for the conducting parts of the antenna using a
c     closed form formula obtained from the vector potential
c     formulation
c
      kh1=2.d0*pi*h1
      c=(d*dsin(teta)*dcos(phi)+hh*dcos(teta)+
     +   l*dsin(teta)*dsin(phi))*2.d0*pi
      b=(dcos(alp)*dsin(teta)*dcos(phi)+dsin(alp)*dcos(teta))*2.d0*pi
      a=dcos(alp)*dcos(teta)-dsin(alp)*dsin(teta)*dcos(phi)
      p1=(0.d0,1.d0)
      a2=2.d0*pi*a
      q1=i1*(dcos(a2*h1)-dcos(kh1)+p1*dsin(a2*h1)-p1*a*dsin(kh1))
      q2=cdexp(p1*a2*h1)*(p1*a*dsin(kh1)-dcos(kh1))
      if ((a.ne.(-1.d0)).and.(a.ne.(1.d0))) then
      q1=q1/((1.d0-a**2)*dsin(kh1))
      q2=i2*(q2+1.d0)/((1.d0-a**2)*dsin(kh1))
      else
      q1=-kh1*dsin(a*kh1)+p1*kh1*dcos(a*kh1)-p1*dsin(kh1)
      q1=i1*q1/(-2.d0*a*dsin(kh1))
      q2=-kh1*a*dsin(kh1)-p1*kh1*dcos(kh1)+p1*dsin(kh1)
      q2=i2*q2*cdexp(p1*a*kh1)/(-2.d0*a*dsin(kh1))
      endif
      aa=cdexp(p1*c)
      c2=1.d0
      if ((b*w1).ne.(0.d0)) then
      c2=c2*dsin(w1*b)/(w1*b)
      endif
      aa=aa*c2*(q1+q2)*r
      ete=aa*(-dcos(teta)*dcos(phi)*dsin(alp)-dcos(alp)*dsin(teta))
   69 continue
      eph=aa*(dsin(phi)*dsin(alp))
c
      return
      end
```